DOCTOR OF PHILOSOPHY

Sequential Recognition of Manipulation Actions Using Superpixel Group Mining

Huang, Tianjun

Award date:
2019

University of Dundee

Doctor of Philosophy

Sequential Recognition of Manipulation Actions Using Superpixel Group Mining

---

Author:

Tianjun Huang

Supervisor:

Prof. Stephen McKenna

Dr. Jianguo Zhang

Computer Vision and Image Processing Group

Computing

School of Science and Engineering

July 2019

Declaration of Authorship

Candidate's Declaration

I, Tianjun Huang, hereby declare that I am the author of this thesis; that I have consulted all references cited; that I have done all the work recorded by this thesis; and that it has not been previously accepted for a degree.

Signed: _____

Date: 01/10/2019

Supervisor's Declaration

I, Stephen J. McKenna, hereby declare that I am the supervisor of the candidate, and that the conditions of the relevant Ordinance and Regulations have been fulfilled.

Signed: _____

Date: 01 October 2019

Acknowledgements

First, I would like to thank my supervisor, Prof. Stephen J. McKenna, for his great supervision and guidance throughout my research. He teaches me how to think independently and critically. Thanks for his encouragement and patience, which really support me a lot to finish my research.

I would like to thank Dr. Jianguo Zhang who also helped me a lot in study and life.

I would like to thank Prof. Emanuele Trucco and all the members in our Computer Vision and Image Processing group for the research discussions throughout my PhD. It is really a pleasant time in the group. We discuss, solve problems, and have fun together.

I also would like to thank my family and my friends for giving me confidence and courage.

The research presented in this thesis is funded by Scottish Informatics & Computer Science Alliance (SICSA) through a prize studentship, and funded by the Computing in University of Dundee. Many thanks for the support and help.

List of Publications

Sequential Recognition of Manipulation Actions Using Discriminative Superpixel Group Mining

Tianjun Huang, Stephen McKenna, in IEEE International Conference on Image Processing (ICIP), 2018

Superpixel Group Mining for Manipulation Action Recognition

Tianjun Huang, Stephen McKenna, in ReaLX 2018: Proceedings of the SICSA Workshop on Reasoning, Learning and Explainability, 2018

Other Publications During PhD:

Detecting and Segmenting Nanodiscs in Immuno-Electron Micrographs

Tianjun Huang, Christian Hacker, John Lucocq, Stephen McKenna, in Medical Image Understanding and Analysis (MIUA), 2014

Hydrodynamic Stretching for Prostate Cancer Detection

Yuri Belotti, Michael Conneely, Scott Palmer, **Tianjun Huang**, Paul Campbell, Stephen McKenna, Ghulam Nabi, David McGloin, in Bio-MEMS and Medical Microdevices II, 2015

High-Throughput, Imaging based Mechanical Phenotyping of Prostate Cancer Cells

Yuri Belotti, **Tianjun Huang**, Stephen McKenna, Ghulam Nabi, David McGloin, in Conference on Lasers and Electro-Optics Europe & European Quantum Electronics Conference (CLEO/Europe-EQEC), 2017

High-Throughput, Time-Resolved Mechanical Phenotyping of Prostate Cancer Cells

Yuri Belotti, Serenella Tolomeo, Michael J Conneely, **Tianjun Huang**, Stephen J McKenna, Ghulam Nabi, David McGloin, in Scientific Reports Journal, 2019

Abstract

Human action recognition is one of the important research areas in computer vision. Manipulation action recognition which contains complex human-object interactions is a challenging problem in this area. Especially for the manipulation actions in the kitchen scenario, occlusions among objects and human body parts and the transformations of objects increase the difficulty for action recognition.

Previous methods on manipulation action recognition often rely on high-level representation approaches such as object detection and human body part detection, which contain expensive work of object and human annotations. Meanwhile, the object transformation information has not been considered comprehensively.

This thesis proposes a method for manipulation action recognition by generating and mining superpixel groups in the videos. Manual annotations of objects and human body parts are not required in the method. We develop a new mid-level representation method called superpixel groups to capture object parts, human body parts and object transformation information in manipulation actions. A hierarchical structure can be built based on the superpixel groups. A participant-based mining algorithm is introduced in this thesis. The proposed mining approach combines discriminativity and representativity characteristics to retrieve discriminative patterns for each action, which is more effective than mining methods that only use discriminativity characteristic.

We evaluate the proposed method on two challenging manipulation action datasets, achieving state-of-the-art result on the 10-class classification problem in the 50 Salads dataset in terms of frame-wise accuracy. Our method also obtains comparable result with the methods using additional object detection and human skin detection in the "Actions for Cooking Eggs" dataset contest.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Manipulation Actions

Human action recognition is a basic research area in computer vision. The purpose of this recognition is to automatically detect and label actions from video [3]. It has wide applications such as human-computer interaction, cognitive situational support, surveillance, motion capture and animation, unconstrained video search and so forth. Manipulation actions refer to the actions where people manipulate objects. It often contains rich human-object interactions. The recognition of manipulation actions is one of the challenging problems in action recognition.

Manipulation actions are tightly associated with our daily living. Human-object interactions happen all the time in everyday work and study. Therefore, the applications of manipulation action recognition can be integrated into many aspects of our lives. For instance, for children, elders and people who may need cognitive situational support, manipulation action recognition systems could form components in an assistant to look after their daily living and seek help if accidents happen [40, 79]. For robotic technologies, manipulation action recognition is often used as the first step for robots to help them manipulate objects [109].

Compared with general human action recognition problem, the situation in manipulation actions is more complicated. Figure 1.1 shows an example for general

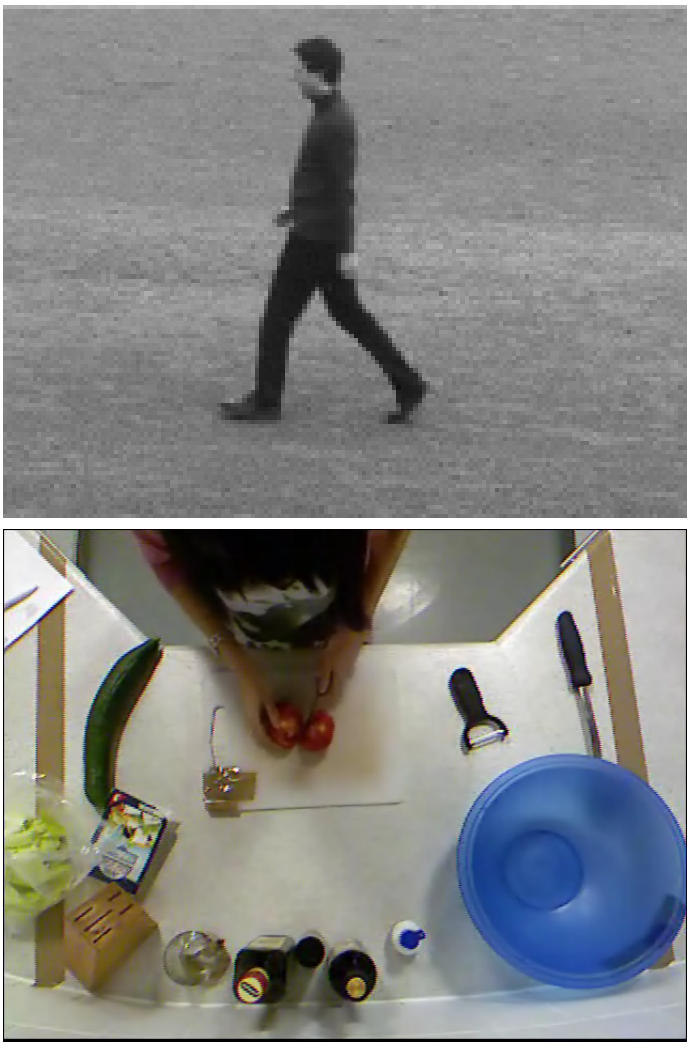


FIGURE 1.1: The top is an example for action "walking" in the KTH dataset [84]. The bottom is an example for action "cut ingredient" in the 50 Salads dataset [89].

action "walking" in the KTH dataset [84] and an example for manipulation action "cut ingredient" in the 50 Salads dataset [89]. One of the vital representative characteristics of manipulation actions is that they usually have low inter-class variability in terms of human motions, which means human motions may be similar among different manipulation actions. Many traditional human action recognition methods [3] which only focus on classifying the actions that have large variability between each other (e.g., swimming and walking) are not appropriate to solve the manipulation action recognition problems (e.g., cutting and peeling). Because of the low inter-class variability, manipulation actions are sometimes referred to as fine-grained actions in the literature [80].

In manipulation actions, much attention has been paid to the human-object interactions in the kitchen scenario [80, 85, 89]. This scenario is close to our daily life and it contains many complex manipulation actions. Occlusions between human and objects, and between objects and objects, are quite common in this scenario. This creates many obstacles for object detection and tracking, and human detection and tracking. Another important problem is that the shape and topology of objects can be markedly changed by manipulation actions in the kitchen. For instance, food ingredients may be divided into several parts and different ingredients can be merged together. These situations increase the difficulty of the recognition task. We will evaluate our proposed method with two kitchen datasets in Chapter 7.

1.2 Limitations in Previous Work

Researchers have made various attempts at recognizing manipulation actions. In order to extract human motion information and object information during human-object interactions, many of them used high-level representations to track human body parts and objects [71, 77, 103, 117]. For example, Prest et al. [77] applied people detection, object detection and tracking methods, and then combined these together to classify human-object interactions. Wei et al. [103] generated a 4D human-object interaction model which applied human pose detection to help solve action recognition problems. However, for human pose based methods, it will be hard to generate pose estimation when only parts of human body occur in the view. In addition, expensive manual annotations are often required in these methods to build detectors for objects and human body parts. Annotations are very time-consuming work, especially when there are lots of objects in the dataset. Other methods utilized low-level and mid-level representation approaches to recognize manipulation actions [90, 116]. Manual annotations for object detection or human body part detection are not required in these methods. However, information is missing in these approaches, which is the object transformation. The shape and topology of manipulated objects can be markedly altered such as those involved

in cooking actions (e.g., cutting into pieces and mixing ingredients). The previous methods often assume that objects in human-object interactions do not have this change, so that they can smoothly track objects and human body parts in videos. They usually apply traditional spatio-temporal tube methods to capture mid-level representation information, which cannot well handle the issue of object transformations. In recent years, deep learning is getting more and more attention. Ni et al. [72] proposed a network which uses convolutional neural networks (CNN) and long-short term memory (LSTM) recurrent neural networks to recognize manipulation actions, but it needs manual annotations for object detection as well. To explore the object transformation and alleviate manual annotation work, in this thesis we propose a new method for recognizing complex manipulation actions in daily living activities. The proposed unsupervised superpixel grouping algorithm does not require manual annotations of objects and human body parts, and contains object transformation information.

1.3 Goals and Method Overview

The aim of this thesis is to provide a new method for recognizing complex manipulation actions in daily living activities. There are three main goals:

- Manual annotations in high-level representation methods for recognizing manipulation actions are quite time-consuming and sometimes may be not feasible in reality. Therefore, one goal of our method is that object detection and human detection are not required in the system. The necessary supervised information is just the action label for each video frame.

- The second goal of the method is to generate representations which can include object transformation information in manipulation actions. This is useful to deal with the change of object shape and topology accompanying human-object interactions.

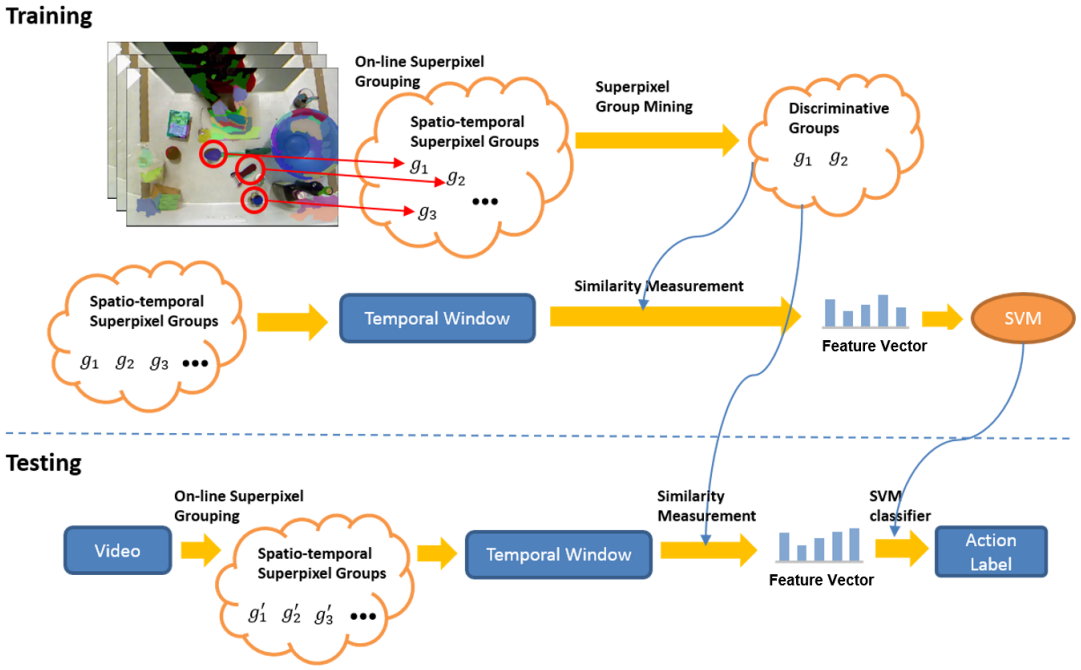


FIGURE 1.2: System overview.

- Furthermore, this recognition system should have the potential ability to be applied in online real-time monitoring applications.

Figure 1.2 shows the overview of our proposed method. Each video frame is first over-segmented into small superpixel regions. Each superpixel is then assigned a foreground probability. Foreground superpixels are selected by setting a threshold on the foreground probabilities of superpixels. We connect these foreground superpixels spatially and temporally in an online manner to generate spatio-temporal superpixel groups. These groups include structures such as bifurcations and loops, which is different from conventional spatio-temporal tubes [62, 74, 96]. These groups are able to represent object transformations such as division and merge.

In the training process, we propose a participant-based mining algorithm which combines discriminativity and representativity characteristics to retrieve discriminative superpixel groups for each action class. Instead of directly recognizing manipulation actions in each single video frame, we use a traditional sliding temporal window approach to predict action labels for video frames. The discriminative superpixel groups are compared with all superpixel groups in the temporal window to build a feature vector representation for this temporal window. These

feature vectors are used to train a multi-class SVM classifier for manipulation action recognition. In the testing process, we use the same strategy to generate spatio-temporal superpixel groups in videos. The obtained discriminative superpixel groups in training process are applied on the temporal windows of testing videos to build feature vector representations. Finally, the trained SVM classifier predicts action labels for each temporal window. We also propose a hierarchical superpixel group structure which combines spatio-temporal groups so as to extract information of multiple associated objects in human-object interactions.

We evaluate our algorithms on two manipulation action datasets: the 50 Salads dataset [89] and the Actions for Cooking Eggs dataset [85]. Both of them are public datasets and different methods have been evaluated on these two datasets in previous works. Results show that our proposed method achieves state-of-the-art results on the 50 Salads dataset in terms of the 10-class classification problem. Our result is better than recent methods using deep networks on this dataset in terms of frame-wise accuracy [54, 55, 115]. We obtained a good performance on Actions for Cooking Eggs contest dataset as well. Our method performs better than methods using local features for training. Furthermore, our results are comparable with methods using object detection and human skin detection.

1.4 Contributions

Our proposed system used the traditional sliding temporal window approach. The duration of each temporal window is about five seconds. Therefore, it is able to be applied on online real-time monitoring applications. Expensive manual annotations for object detection or human body part detection are not required in our method. It can save computational time when there are lots of objects in the training set. This advantage also makes the system be possibly applied on robotic technologies which the actions are manipulated by robots rather than human beings.

Firstly, we have contributions for mid-level representation in manipulation action recognition:

- We propose superpixel groups able to capture fine-grained motions in manipulation actions. They can be used to model object transformations which are often missing in previous methods during human-object interactions.
- Based on our superpixel groups, we built a hierarchical superpixel group structure. This structure can contain multiple human body parts and object parts, which has potential ability for providing more human-object interaction information.
- We propose a participant-based mining method which combines discriminativity and representativity characteristics for searching discriminative patterns of each action class. The result shows that our method has a better performance than the method which only considers discriminativity characteristic.

Secondly, we obtain a good performance on two public datasets compared with previous methods:

- We achieve state-of-the-art results on 50 Salads dataset compared with other methods. We also obtain better performance than recent work using deep networks [54, 55, 115] in terms of frame-wise accuracy. Meanwhile, compared with deep network methods, our algorithm uses short-range temporal information only. Furthermore, the model proposed in our thesis has a directly interpretable representation. Researchers can easily check the system performance and tune corresponding parameters by looking at retrieved discriminative superpixel groups, while model selection and tuning parameters may need more work for deep network methods.
- We also obtain a good performance on another manipulation dataset, Actions for Cooking Eggs. Our performance is better than the methods which

also only use action labels of video frames for training in the contest. In addition, our method is comparable with the methods using additional object detection and human skin detection information in the Actions for Cooking Eggs dataset contest.

1.5 Thesis Structure

- Chapter 2 reviews relevant related work on manipulation action recognition. We separate previous work into four categories: low-level representation methods, mid-level representation methods, high-level representation methods and deep learning methods.

- Chapter 3 introduces our foreground superpixel segmentation method. We define foreground superpixels as the superpixel regions which are clearly above the work surface. Everything else is considered as background.

- Chapter 4 proposes an algorithm called spatio-temporal superpixel grouping. The superpixel group is a mid-level representation for manipulation action recognition. This chapter demonstrates how to build spatio-temporal superpixel groups in an online way in videos.

- Chapter 5 investigates a new structure called hierarchical superpixel groups. We build this structure upon our superpixel groups.

- Chapter 6 details the proposed mining algorithm for selecting discriminative superpixel groups. The concepts of discriminativity and representativity are discussed. Then, we describe our participant-based mining method which combines these two characteristics. In addition, this chapter shows the construction of representations for sliding temporal windows based on the discriminative superpixel groups. A classification method is discussed in this chapter as well.

- Chapter 7 reports evaluation results of the proposed methods on the 50 Salads and the Actions for Cooking Eggs datasets. We also compare performance with other approaches on these two datasets.

- Chapter 8 summarizes our contributions and recommendations for future research.

Chapter 2

Literature Review

There is a large body of literature on the human action recognition problem. In this project, we focus on the manipulation action recognition problem. The main difference between manipulation action recognition and general action recognition is that human motion information is usually insufficient to accomplish the recognition task; associated objects have to be considered. On the one hand, object information may be required in the action classification (e.g., cutting tomatoes or cutting cucumbers). On the other hand, since some human motions are very similar to each other (e.g., cutting and peeling), object information together with the associated human motion can provide another cue to classify the actions. Therefore, how to combine human motion information and object information is the critical question in manipulation action recognition.

Here we focus on recognising manipulation actions observed using a static camera. Manipulation recognition tasks have also been analysed in the context of egocentric video where the camera is moving; the focus has mainly been on how to predict and track gaze and attention (e.g., [17, 59, 106]).

We separate the most commonly used methods in manipulation action recognition into four categories: low-level representation methods, mid-level representation methods, high-level representation methods and deep learning methods. Low-level representation methods usually extract low-level image features (e.g., SIFT

[27], HOG [15], HOF [10]) and then encode these features with Bag of Words [58] or Fisher vectors [75] to represent an action sequence. Compared with low-level representation, mid-level representation is usually a spatio-temporal sub-region of the video. Rather than directly extracting low-level features from local image pixels, mid-level methods are trying to extract features from these spatio-temporal 3D regions. These features are then utilized to form a representation of an action sequence for final action recognition (e.g., [102, 114, 116]). High-level representation methods refer to the approaches that use representations which can be interpreted by humans, such as the methods using human pose [103] or object detection [69] to assist to recognize actions. Deep learning methods apply deep neural networks (e.g., [86]) on the video frames followed by recurrent neural networks [30] to predict action labels.

2.1 Low-level Representation Methods

The general steps for the low-level representation approaches are: (i) interest points are detected; (ii) the feature descriptors are generated based on these interest points; (iii) these descriptors are assigned to a set of feature vocabularies which can be used in a bag-of-words representation for a video sequence [52, 64, 100]; (iv) finally a classifier (e.g., SVM) is learned for the action recognition.

In an important paper, Wang et al. [100] densely sampled interest points at multiple scales in each video frame and tracked them by using dense optical flow. They generated a variety of features at the interest points (i.e., trajectory shape, histograms of oriented gradient (HOG), histograms of optical flow (HOF), motion boundary histograms (MBH) [16]). These features were then clustered by k-means and represented by bags of features. SVM classifier was used for action classification. Experiments indicated that this method improved the recognition accuracy of previous work on nine commonly used datasets. Furthermore, by trying to separate the camera motion and the human motion, Wang et al. [101] improved their results again on four datasets. Table 2.1 shows the recognition accuracies of the

dense trajectories method [100] and improved trajectories method [101] on four datasets.

Datasets	Dense trajectories [100]	Improved trajectories [101]
Hollywood2 [68]	60.1%	64.3%
HMDB51 [50]	52.2%	57.2%
Olympic Sports [73]	84.7%	91.1%
UCF50 [47]	88.6%	91.2%

TABLE 2.1: Comparison of the dense trajectories method and improved trajectories method on four datasets.

The low-level representation is not rare to see in manipulation action recognition. Rohrbach et al. [80] proposed a manipulation dataset of cooking activities. This dataset contains fine grained actions between human and objects in the kitchen (e.g., pour, cut slices). They evaluated multiple low-level features: trajectory, HOG, HOF, MBH and their combination on this dataset. These features were then encoded with Bag of Words to train SVM classifiers. The recognition accuracy became a benchmark on this dataset. Stein and McKenna [89] created a manipulation dataset called "50 Salads". It contains 50 videos capturing people making mixed salads. Human-object interactions happen frequently in this dataset. In their experiments, they extracted various low-level visual features (e.g., Absolute Tracklets, HOG, HOF, MBH) from the dataset. These features were encoded with Bag of Words and used to train SVM classifiers for action recognition. A *sliding temporal window* approach is usually used in these methods to classify and segment manipulation actions.

Instead of using sliding window to recognize actions, another thought is to apply *structured temporal models*. Rather than classifying each temporal window, the whole video can be an input. These methods recognize manipulation actions by finding a most probable sequence of actions. Kuehne et al. [49] proposed a generative framework for video segmentation and recognition. They also used dense trajectories to extract features. The dimensionality of the features was reduced by using Principal Components Analysis (PCA). A set of Hidden Markov Models (HMM) were utilized to model all action units in the corresponding dataset. They segmented and recognized actions by applying Viterbi algorithm to search

the most probable sequence of action units. Richard and Gall [78] combined three models: language model, length model and action model to segment the input video into a number of segments and each segment is assigned with an action class label. They extracted improved dense trajectories to generate features and then encoded with Fisher vectors. The final maximization problem was solved by using dynamic programming.

The low-level representation methods have been commonly used in action recognition. Low-level features followed by global pooling approaches mainly focus on capturing information of global motions in the videos. However, in manipulation actions, there are many fine-grained motions. The low-level representation methods can easily be affected by the background movement and miss the important human-object interaction movement. The low-level information in the low-level features limits their discriminative ability for high-level motion recognition.

2.2 High-level Representation Methods

The high-level representation methods which we mention here utilize interpretable semantic representations to help to recognize manipulation actions. These representations can be easily understood by everybody. Human pose and object detection are two commonly used approaches in manipulation action recognition.

The structure of the skeleton of the human body determines the human motion patterns. Johansson [45] showed that, by tracking markers of the main joints of the human body, the human motion patterns can be recognized from the motions of these spots (Figure 2.1). This early work motivated many later methods to use the motions of landmarks on the human body to represent human actions. The features of human body joints can be extracted based on their spatial and temporal information, and then sent to a classifier to train a model for the action recognition [33, 99]. Sometimes it is not necessary to track the whole human body; a coarse body representation is enough to accomplish the recognition task, e.g., trajectories of the upper body and hands ([34]).

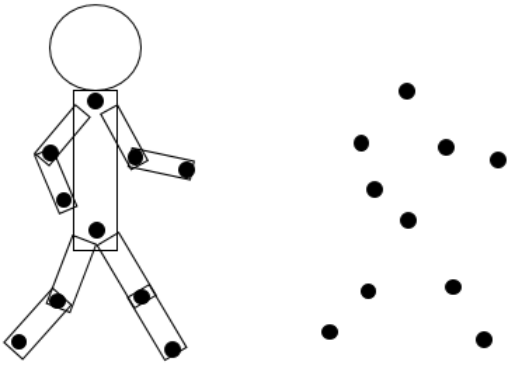


FIGURE 2.1: Example of bright spots on human body. The left is a walking person and the right is the corresponding pattern of spots on joints.

Since manipulation actions usually happen between human and objects, detecting the classes of objects is a reasonable and obvious way to assist manipulation action recognition. Due to the attributes of objects, some actions may only appear on specific objects and vice versa.

Packer et al. [76] combined a pose tracker method [28] with object detection to recognize 13 actions in a cooking dataset and the VISINT [1] dataset which contains action videos in outdoor scenes. Prest et al. [77] leveraged the best human detection, object detection and tracking methods at that time. The 3DHOG-track descriptor [48] and an interaction descriptor which contains the relative location, area and motion between human and objects were introduced and applied in their method. In the classification, they trained a linear SVM by using the outputs of three classifiers. Rohrbach et al. [81] created a new video dataset called MPII Cooking 2. Manipulation actions in this dataset look visually very similar with each other and have low inter-class variabilities. A new pose estimation approach was proposed by them based on the work of Andriluka et al. [7]. They also developed a hand detector based on the work of Felzenszwalb et al. [23]. Both the pose-based method and the hand-centric method were evaluated on their dataset to recognize manipulation actions. Ping et al. [103] built a 4D human-object interaction model learned from RGB-D videos which uses the human pose as input to solve action recognition and segmentation problems. Other works combined

hand detection and object detection (e.g., [69–71, 117]), which are also used in manipulation action recognition.

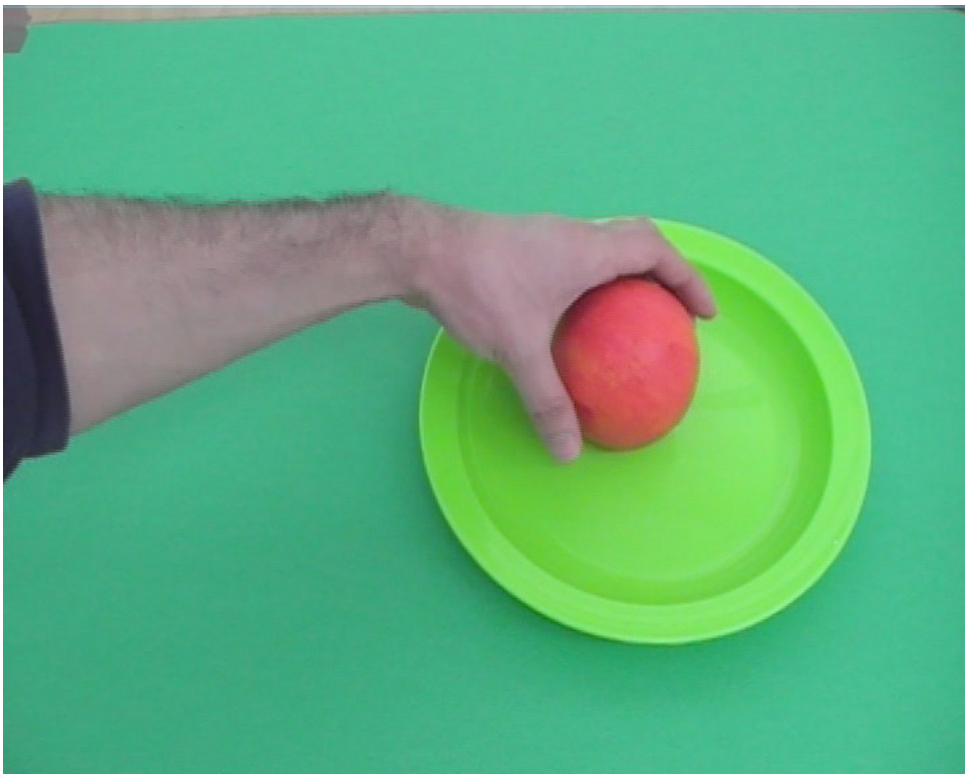


FIGURE 2.2: A video frame example in the Manipulation Action Consequences dataset [108].

However, human-pose based methods, object detection based methods and their combinations all have shortcomings in applications. For human-pose based methods, in real life, in many cases cameras only can capture the images with partial human body. These images or videos are not enough for common algorithms to build a human pose model. Moreover, tracking the human body in complex environments is a challenging task. The change of viewpoints and a variety of occlusions increase the difficulty of human pose estimation. For object detection based methods, in order to build the object detectors, manual annotations of each kind of object in the dataset are required. This is very expensive work especially when there are lots of objects in the dataset. Meanwhile, frequent occlusions between human and objects, and between objects and objects present great obstacles for object detections. In the literature, there are some works which try to use unsupervised approaches to track and detect objects in manipulation actions. Yang et al. [108] utilized an unsupervised tracking method to locate objects. However,

this was against a clear background (Figure 2.2). Aksoy et al. [5] also used unsupervised methods to track objects in a simple dataset, but on realistic cooking datasets manual annotations were still required to build object detectors.

2.3 Mid-level Representation Methods

Compared with low-level and high-level methods, mid-level representation focuses on spatio-temporal regions in the videos. These 3D regions contain more appearance and motion information than the features obtained from low-level interest points. These regions are often generated by using unsupervised algorithms, so that expensive manual annotations for objects are avoided. Figure 2.3 shows an example of 3D sub-regions in a video sequence.

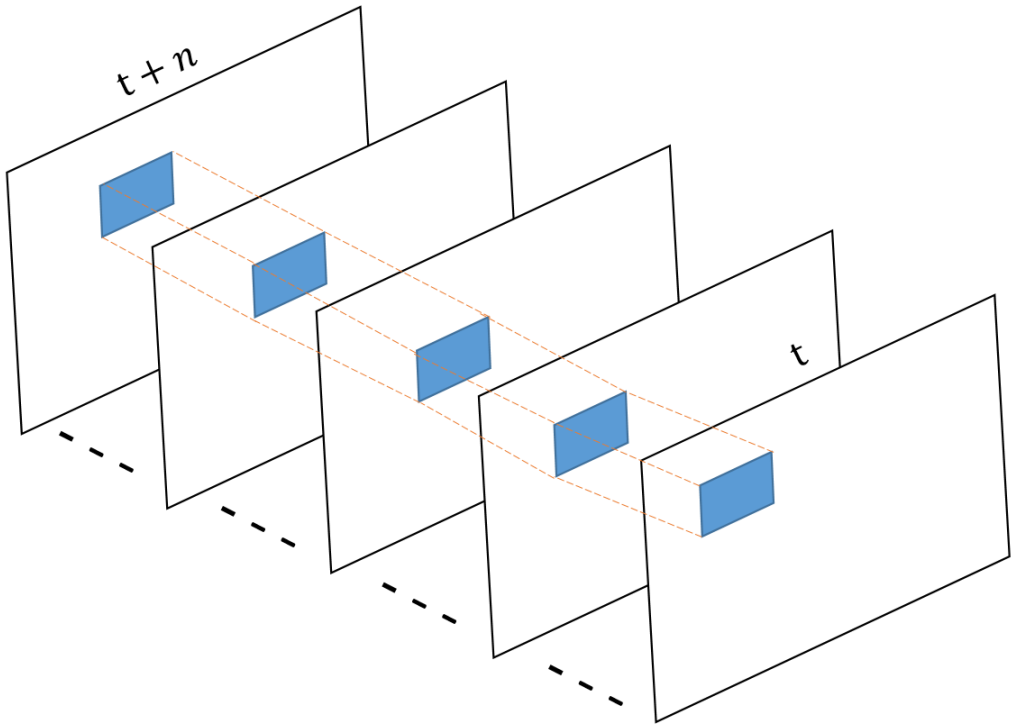


FIGURE 2.3: An example of spatio-temporal tube in a video sequence from frame t to frame $t + n$.

For the mid-level representation methods in video, there are usually three main ideas:

- The first is to use *discriminative patches* followed by a max pooling on the spatial pyramid [11, 43, 87]. In these methods, a large number of spatio-temporal patches in the videos are generated first. Since it is not necessary to process all the patches and the computation is expensive as well, a set of discriminative patches are selected to be the detectors. To perform action recognition, max pooling on the detector scores can be used to form a spatial pyramid structure, which is also called volumetric max-pooling [82]. These traditional mid-level methods need to divide a video using fixed grids. However, space-time invariance is missing during the process. Human and objects may appear at different locations in videos. Pre-grid segmentation may cause their representations to be inconsistent.

- The second idea is to use *supervoxels*. This extends method from 2D superpixels by adding the temporal information to build a 3D segmentation. Jain et al. [44] utilized a graph-based method [105] to form the initial supervoxel segmentation. These supervoxels were then iteratively merged to create new supervoxels. Each supervoxel was represented by bag of words. These representations were then used to train a classifier for each action class. Oneata et al. [74] applied SLIC [2] to oversegment each video frame. These SLIC superpixels were merged spatially and temporally to form hierarchical supervoxels. They used three measurements to connect superpixels: spatial neighbor connections, second-order spatial connections and temporal neighbor connections. The hierarchical clustering was completed by applying average linkage [8]. Van Den Bergh et al. [96] proposed an online supervoxel algorithm based on their 2D superpixel segmentation method. Yi et al. [112] generated a content-sensitive supervoxel method which outperforms existing supervoxel methods in their experiment.

- The third idea is to generate *object proposals* (e.g., [53, 93]) in each video frame first. These object proposals can provide many sub-regions which possibly contain objects in the image. Focusing on the proposals can reduce computational time and alleviate the influence of unimportant background. The objects mentioned here can include hands or other human body parts

as well. In order to fuse temporal information in videos, the proposals are linked in the temporal domain to build spatio-temporal regions. Lan et al. [51] generated region proposals in each video frame using the method of [18]. Spatio-temporal segments were obtained by performing spectral clustering based on the similarities between pairs of spatial region proposals. They built a spatio-temporal segmentation tree for each video. A graphical model was learned from these trees with their mid-level action elements to recognize and parse the actions in the new videos. Their method has been evaluated on a manipulation cooking dataset and also other general action datasets to show its ability. Zhou et al. [116] developed a mid-level approach to recognize manipulation actions. They used Binarized Normed Gradients (BING) [13] to generate object proposals. These proposals were then merged spatially and temporally based on their spatial overlap, trajectory link strength and appearance similarity to form sapatio-temporal regions. They selected discriminative regions for each manipulation action class using the method of [46]. A max-N pooling approach was proposed by them to generate a feature vector as the video representation. LibLinear [20] was trained as the classifier.

The mid-level representation methods in the literature often assume that the topology and shape of objects in the videos will not change a lot. Therefore, a generated spatio-temporal tube is only a single path. However, manipulation actions often involve frequent object-object and human-object occlusions and complex changes in shape, and topology. Object transformation information in the previous methods [51, 116] has not been considered comprehensively. Actions like cutting something into pieces or mixing ingredients can change the object topology and shape markedly. A more comprehensive structure needs to be created to contain object transformation information in manipulation action recognition.

2.4 Deep Learning Methods

Deep learning (DL) methods have become popular in recent years and have advanced the state-of-the-art results on many datasets. DL is a multi-layer neural network structure. The outputs from multiple layers can be thought of as the progress from low-level representation to high-level representation. Therefore, we separate DL out from the previous three sections and discuss its recent applications to manipulation action recognition in this section.

Lea et al. [55] proposed a spatio-temporal CNN model to recognize fine-grained manipulation actions. The spatial component was based on VGG [86] and the temporal component was computed by applying a 1D convolution between temporal filters and a sequence of spatial features. A Semi-Markov Conditional Random Field [83] was used to jointly segment and classify actions. Lea et al. [54] further developed two Temporal Convolutional Networks (TCNs), Encoder-Decoder TCN and Dilated TCN, for manipulation action recognition. The Encoder-Decoder TCN hierarchically built the network with temporal convolutions, pooling and upsampling. The Dilated TCN was an adaptation from the method of WaveNet [97]. It uses dilated convolutions and has skip connections between layers. Lei and Todorovic [57] developed a temporal deformable residual network for manipulation action recognition. The input of the network was frame-level CNN features and the output was their corresponding action labels. Their model has two streams: residual stream and pooling/unpooling stream. They proposed a deformable temporal residual module to combine these two streams together to form the whole network. Zhang et al. [115] also utilized a temporal convolutional encoder-decoder net combined with a proposed bilinear pooling operation to recognize manipulation actions.

The inputs of all above mentioned deep learning networks are video frames and the outputs are their action labels. However, deep learning methods can also be combined with other low-level, middle-level or high-level representation methods which we mentioned before to recognize manipulation actions. For instance, Ni et al. [72] proposed a network to detect objects in the actions first and then

recognize actions based on the detected objects. Their method was inspired by the success of long-short term memory (LSTM) [38] recurrent neural networks [30]. The novelty here was that they applied a set of LSTM nodes to incrementally refine the object detection results. After the object detection, they extracted local motion features in the parsed body parts and objects. These features were encoded with improved Fisher vector to generate representations for each object of interest. They concatenated these representations as the feature vector of the video segment and then used them to train linear SVM classifiers to predict action labels.

Although deep learning is popular, it has shortcomings. For instance, deep learning requires a large amount of data to train the model. Meanwhile, the training process can be quite time-consuming and needs powerful GPUs. People often spend quite a lot time tuning the architecture parameters of deep networks when applying them on a new application.

In this thesis, we propose a mid-level representation method with traditional SVM classifier to solve the manipulation action recognition problem. Our method does not require manual annotations of objects. It is able to capture the object transformation information and it has a directly interpretable representation which is helpful when tuning the parameters on other applications.

Chapter 3

Foreground Superpixel Segmentation

3.1 Motivation and Overview

Manipulation actions contain both human motion information and interacted object information. The characteristics of human body parts which are manipulating and the associated objects should be considered. Splitting the video scene into foreground regions (human body parts and objects) and background regions is helpful to save computational time by focusing on the foreground regions. The first step in our method is to extract foreground regions which maybe contain human body parts and interacted objects from each video frame.

We use the 50 Salads dataset [89] as the example to show how our algorithm works (Figure 3.1). The depth information provided in this dataset is used to find foreground superpixel regions. It should be noted that we do not use depth for further feature extraction and manipulation action classification. The static spatio-temporal segmentations will be removed based on optical flow (see Section 4.3.2). Therefore, depth information used in this thesis will not improve the recognition results. The main purpose of depth information is to reduce computational

FIGURE 3.1: A sample of the scene in the 50 Salads dataset [89].

time. Other foreground segmentation algorithms like Gaussian mixture model [56, 119] can be another choice if depth is not provided.

In terms of the categories of actions in manipulation action datasets, the manipulated objects are usually on the work surface. We define everything above the work surface (i.e., food ingredients, tools, containers and hands and arms) as the foreground objects. The work surface itself and everything below it are the background.

In this chapter, we over-segment each video frame into superpixels. Thus, objects are typically divided into multiple small regions. These superpixels enable small parts (e.g., resulting from cutting a large object) to be represented. Random Sample Consensus (RANSAC) [26] is applied on the depth map to locate the plane of the work surface. The foreground probability of each superpixel is computed by using sigmoid function. Foreground superpixels are those whose probabilities are greater than a threshold. The details are introduced in the following sections.

3.2 Foreground Probability Map

From the top-down view in the kitchen, the work surface usually occupies most of the camera view (Figure 3.1). The depth images are provided in the dataset. For each video frame, a 3D space can be created by setting the depth as the z coordinate. Width and height are the other two coordinates (x and y). Then, we adopt the Random Sample Consensus (RANSAC) algorithm [26] to find the plane which represents the work surface in the 3D space.

In order to find the plane, the first step of RANSAC is to randomly sample three points which are not colinear in the 3D space. A plane which contains the three points is created. We define a threshold for the distance between points and the plane. The image points whose distance to the plane are below the threshold are labelled as the points in the plane, also called inliers, otherwise they will be classified as outliers. This threshold is not a sensitive parameter from the view of experiments as long as it is not too large which includes foreground objects. We set the threshold to 0.2 in the experiment. The number of inliers is used as the measurement to indicate the quality of the located plane. The more inliers contained in the plane means that the located plane is more accurate. This process is iterated N times. Each time, the plane which has a greater number of inliers is saved. The final remaining plane after N iterations is the best plane we get for representing the work surface. The number of iterations N can be computed using the following equation [91]:

$$N = \frac{log(1-p)}{log(1-(1-\epsilon)^s)} \tag{3.1}$$

where p is the probability for finding the correct plane, ϵ is the measured percentage of outliers and s is the number of sampled points. The probability p can be set to a high value (e.g., 0.99). After finding the best plane model with RANSAC, there are some fine-tuning methods to improve the current model. The least-squares fit is one of them. The idea in our case is to minimise the difference between the

depth calculated by the plane model function and the real depth captured by the Kinect for the inliers. The equation is shown below:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (f_{\boldsymbol{\theta}}(x_i, y_i) - z_i)^2$$
$$f_{\boldsymbol{\theta}}(x_i, y_i) = \theta_1 \times x_i + \theta_2 \times y_i + \theta_3 \tag{3.2}$$

Where (x_i, y_i, z_i) is the 3D coordinate of a data point in inliers, n is the number of the points and $f_{\boldsymbol{\theta}}$ is the function of the plane.

In the experiment we find that although the work surface is detected successfully in most video frames when we apply RANSAC algorithm on each individual frame, there are still a fraction of wrong plane detections. For instance, the first row in Figure 3.2 shows five consecutive video frames. The red regions in second row indicate the parts above the work surface. They are detected by using RANSAC algorithm for each single frame. As can be seen in the Figure, the paper and cucumber are not included in the red regions for some frames in second row. It demonstrates that the corresponding planes are not detected properly. One of the possible reasons is that the human motion changes the proportion of the work surface in the view so that it increases the difficulty to find the data points on the work surface.

To solve this problem, firstly, we use a small number of frames in the video to find the best plane for the work surface. Since the camera is static, we apply this best plane on all the frames in the video. Specifically, we utilize RANSAC algorithm to detect M work surface planes in the first M frames from the video as the training set. For each frame, the normal vector of the work surface plane and mean depth value of data points in the plane are calculated as the measurement of the work surface plane in this frame. We fit the measurements of M work surface planes to a t-distribution [63] due to the reason that the occurrence frequency of the wrong plane detection is low and the number of frames which we use to find the best

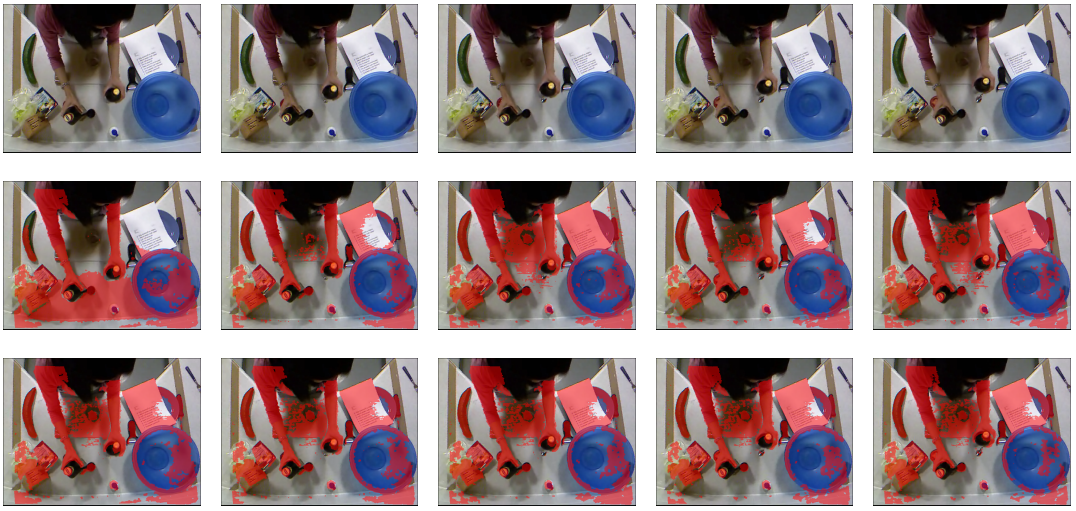


FIGURE 3.2: The first row shows five consecutive frames in a video in the 50 Salads dataset. Red regions in second row are the parts above the plane of the work surface after applying RANSAC algorithm on each individual frame. The third row shows the modified plane detection which uses the first 50 frames in this video to find the best plane model.

plane is small compared with the total number of frames in the video. We choose the measurement which is closest to the mean value of the t-distribution. Its corresponding work surface plane model is the best model. In order to choose the value of M, we randomly select a certain number of frames from training videos and then judge the performance of the detected best plane model by visualizing the regions above the work surface. If the regions contain all the objects above the work surface, it is considered as a good model. We set $M = 50$ based on the experiment (video frame rate is 30 frames per second), which shows enough ability to locate the appropriate plane. The modified plane model can be seen in the third row of Figure 3.2. It alleviates the problem for wrong plane detections of the work surface.

After the plane of work surface is found for the frame, the probability of the image point to be on the foreground (i.e., $P(F|Depth)$) can be directly computed based on the signed distance between this point and the work surface plane. The logistic sigmoid function is used to map the distances to probabilities. Equation (3.3) shows the computation for $P(F|Depth)$:

$$P(F|Depth) = \frac{1}{1 + e^{-D}} \tag{3.3}$$

where D is the signed distance between the data point and the work surface plane.

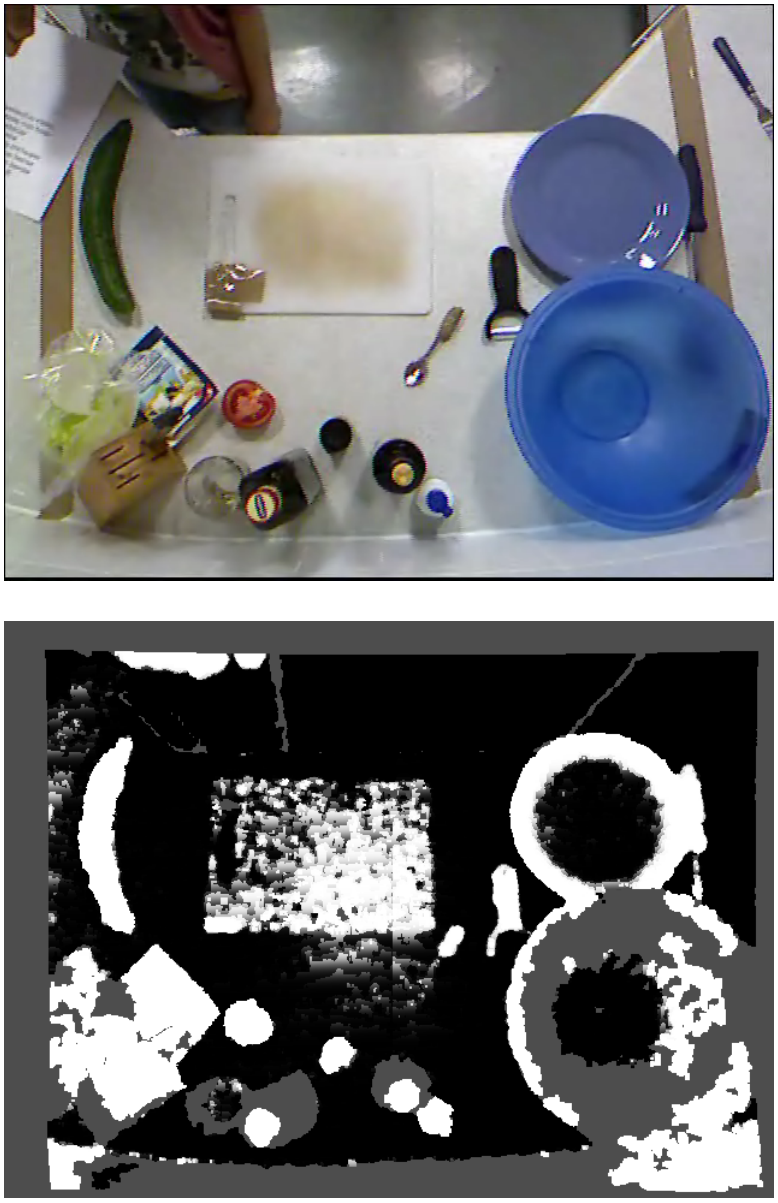


FIGURE 3.3: Top: the original video frame. Bottom: foreground probability map $P(F|Depth)$.

Figure 3.3 shows an example frame and its $P(F|Depth)$ map. The brighter areas have higher probabilities to be a foreground object. As we can see, even though the entire chopping board is hard to detect since its colour and depth are very similar to the work surface, its rough outline and location can still be estimated from

the probability map. In the depth map, some depth values are "Null" generated by Kinect camera. We set the foreground probabilities of these points to a low value (lower than 0.5) to ease their influence to the next stage of processing. The performance of the depth sensor is not perfect so that some information may be missing in the depth map (e.g., the fork in the top image of Figure 3.3).

3.3 Foreground Superpixels

Directly segmenting entire target objects from an image is usually not a trivial task, especially in manipulation action videos where occlusions happen frequently. Superpixel algorithms provide another way to simplify the segmentation process by over-segmenting the image into many small regions. These regions group the pixels to represent parts of the original objects. Features can be extracted from these small regions for further processing. In this thesis, we adopt a fast superpixel segmentation method, Superpixels Extracted via Energy-driven Sampling (SEEDS) [94], to over-segment the video frames. Results in [94] indicate that the SEEDS method is faster than the popular real-time SLIC (Simple Linear Iterative Clustering) [2] method. Furthermore, the performance (e.g., undersegmentation error, boundary recall, etc.) of SEEDS is competitive with the non-real time method: Entropy Rate Superpixels (ERS) [65].

In the SEEDS method, the image is initially partitioned using a regular grid. In each iteration, the previous partitioned region is divided into smaller blocks. The blocks on the boundary of the region try to move to the adjacent regions. The blocks will become smaller and smaller until pixels. Finally, pixel-level tuning of the boundaries of the regions is performed. The movement of blocks is controlled by the energy function [94]:

$$E(s) = H(s) + \gamma G(s) \tag{3.4}$$

where $H(s)$ is a colour distribution term calculated from the colour histograms of superpixels, and $G(s)$ is a boundary term used to control the shape of superpixel boundaries. γ is the weighting factor. The hill-climbing approach is used to find the maximum value for the energy function. In other words, the movements of blocks or pixels on the boundaries of regions are allowed if they can increase the energy. More details of the computation can be found in the SEEDS paper [94].

In order to utilize the depth information, we adopt the Depth SEEDS method [95]. The idea is that, in the pixel-level updates, the movement of a pixel is determined by the distance between its colour and depth and the mean colour and depth of the region which it may be moved to. The pixel is assigned to the region which has the shortest distance D [95]:

$$D = \sqrt{(L - L_s)^2 + (A - A_s)^2 + (B - B_s)^2 + (d - d_s)^2} \tag{3.5}$$

where L, A, B are the LAB colour of the pixel, d is the depth of the pixel, L_s, A_s, B_s, d_s are the mean colour and depth of the region which this pixel may be moved to.

Figure 3.4 shows the SEEDS segmentation result. We hope that the SEEDS segmentation can over-segment video frames, which means that each superpixel only represents a part of an object. In this thesis, each video frame is segmented into 600 superpixels which are enough to over-segment the frame. More superpixels can provide more accurate boundaries for small regions. However, it will increase the computational time for further processing as well.

After obtaining the superpixel segmentation, combining with previous foreground probability map, we are able to calculate the foreground probability $P(S)$ for each superpixel S using Equation (3.6):

$$P(S) = \frac{1}{n} \sum_{i=1}^{n} P(p_i) \tag{3.6}$$

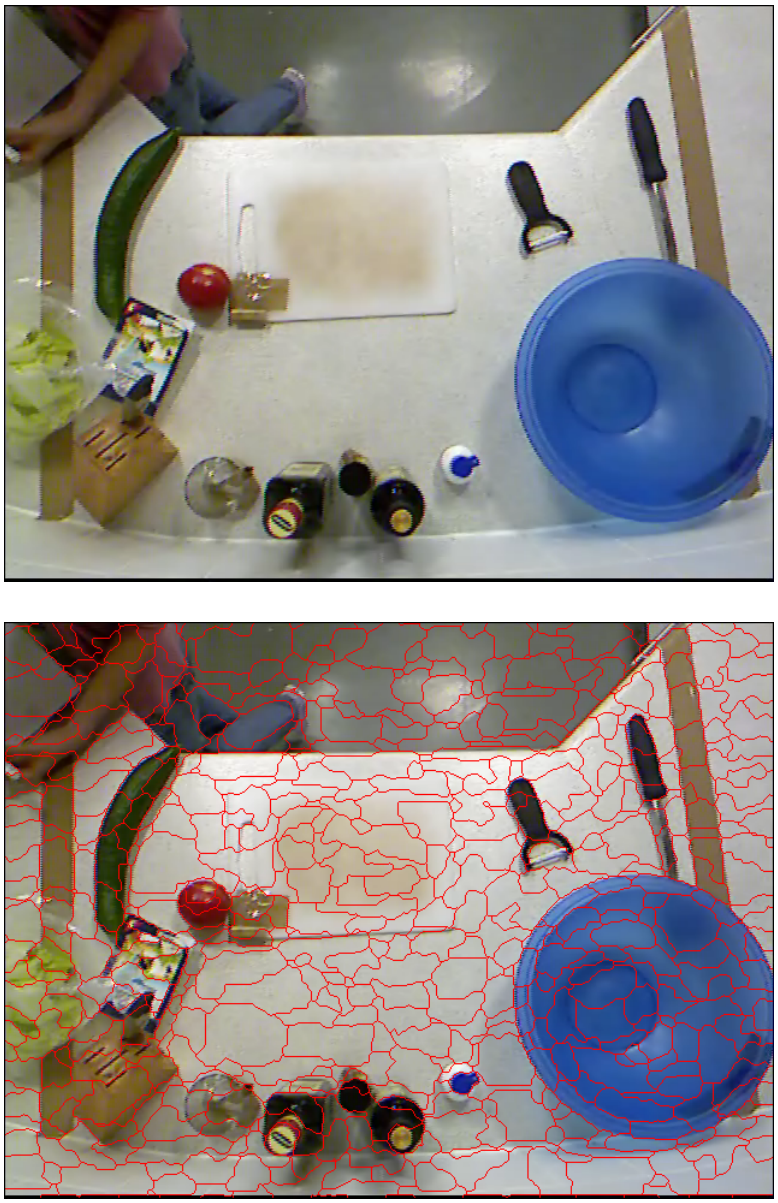


FIGURE 3.4: Top: the original video frame. Bottom: superpixel segmentation on the frame. The red lines are the segmentation boundaries.

where n is the number of points in this superpixel S, $P(p_i)$ which can be computed using Equation (3.3) is the foreground probability of the point p_i in S. Therefore, $P(S)$ is the mean value of foreground probabilities of points in superpixel S. If $P(S) > 0.5$, the superpixel S is determined as a foreground superpixel. Figure 3.5 shows an example of our foreground superpixel segmentation result. These foreground superpixels are utilized to generate our spatio-temporal superpixel groups in the next chapter.

FIGURE 3.5: Foreground superpixel segmentations on a video frame. Each colour region represents one foreground superpixel segmentation.

3.4 Conclusion

In manipulation action recognition, tracking manipulated objects and human body is not a trivial task due to the frequent and complex human-object interactions. Meanwhile, it usually requires expensive manual annotation work to build the detector for each object. Therefore, instead of detecting objects and human body, we use superpixels to represent object parts and human body parts. On one hand, unsupervised superpixel segmentation does not require expensive manual annotations. On the other hand, it is more flexible to deal with the situations during manipulation actions (e.g., one object is divided into several parts or several object parts are mixed together).

We demonstrated one method to segment foreground regions for each video frame in this chapter. We use RANSAC to find the plane of the work surface first. Then, a foreground probability map is generated for each video frame. We combine the probability map with SEEDS superpixel segmentations to search the foreground superpixels. These foreground superpixels are used to represent foreground object parts. It should be noted that the depth information is just applied on the

foreground region segmentation. It will not be used as a feature to recognize manipulation actions.

Chapter 4

Spatio-temporal Superpixel Grouping

4.1 Motivation and Overview

As mentioned in the literature review, compared with low-level representation methods, mid-level representation contains more appearance and motion information; comparing with high-level representation methods, mid-level representation is able to avoid the heavy manual annotation work for building object detectors. Therefore, we use mid-level representation to help to recognize manipulation actions.

The mid-level representation methods in the literature often have an assumption that the topology and shape of objects in the videos will not undergo transformations. For instance, Zhou et al. [116] utilize BING [13] to generate object proposals in each video frame, and then merge the proposals spatially and temporally to establish their interaction parts. The interaction parts which they proposed have a constraint of spatial overlap when connecting object proposals. Jaccard similarity is used to compute the spatial overlap value. When they connect two object bounding boxes in two frames, the function of this spatial overlap measurement is trying to make the two linked boxes similar in size. This constraint is helpful to

track object boxes when the shape of the object is not changed a lot. However, manipulation actions such as those involved in food preparation often markedly change topology (e.g., cutting a vegetable into several pieces). In order to solve this issue and extract object transformation information, based on the foreground superpixels which we obtained from the previous chapter, we propose a spatio-temporal superpixel grouping algorithm.

The idea is that, for each foreground superpixel, we connect it with foreground superpixels around it in the same frame based on their colour similarity, to form spatial superpixel connections. Then, we connect the superpixel with foreground superpixels in the previous frame based on their colour and motion similarities to form temporal superpixel connections. As the video moves forward frame by frame, these connections may separate foreground superpixels into different superpixel groups. These groups are our mid-level representations for manipulation action recognition.

The important difference between our superpixel groups and conventional spatio-temporal tubes [62, 74, 96] is that bifurcations and loops are allowed in our groups. Transformations such as separation and merging in manipulation actions can be represented in our groups. Meanwhile, compared with large bounding boxes, small superpixels are able to capture more fine-grained changes in different actions.

We also set a rule to constrain the temporal length of superpixel groups. The frame rate in the 50 Salads dataset is 30 frames per second. Each video can contain more than 10,000 frames. Without temporal constraint, the superpixel groups can start from the first frame of a video to the end frame. These groups span multiple actions. They are not helpful to distinguish different action classes. Therefore, we develop a grouping threshold that increases as a group lengthens to avoid the occurrence of very long duration superpixel groups.

4.2 Spatial Superpixel Connection

In order to build spatial connections for our superpixel groups, we link the foreground superpixels in the same frame based on their appearance similarity. We generate a colour histogram for each superpixel as in [93]. Specifically, each colour histogram has 75 bins, 25 bins per channel. The appearance similarity $a(s_i, s_j)$ of two superpixels s_i and s_j is measured using colour histogram intersection. The range of the calculated appearance similarity is between zero and one.

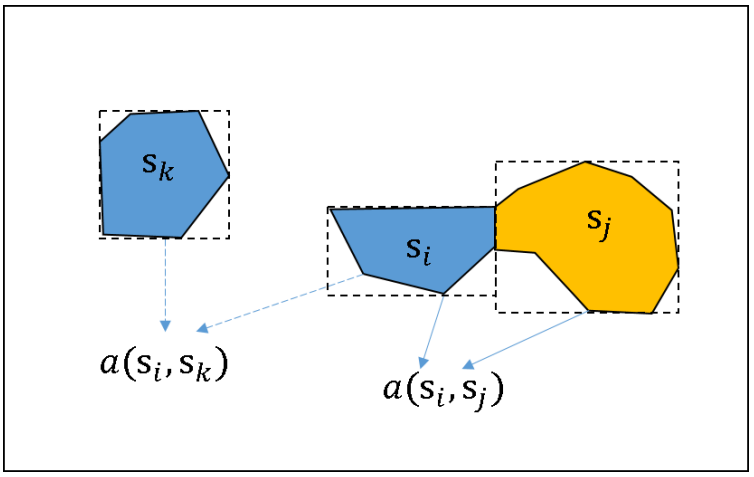


FIGURE 4.1: s_i, s_j, s_k are three foreground superpixels in the same video frame. $a(s_i, s_k)$ will not be computed since the bounding boxes of s_i and s_k are not adjoined or overlapped with each other.

We only compare a superpixel with its adjacent superpixels. Figure 4.1 shows, s_i, s_j, s_k, three foreground superpixels in the same video frame. For each foreground superpixel, we generate a bounding box. The similarity of two superpixels is computed only when the two bounding boxes are adjoined or overlap. Otherwise, the similarity will not be calculated, avoiding the need to compute histogram intersection of these two superpixels.

Since the purpose of spatial connection is to link foreground superpixels which have similar appearance, we set a relatively high threshold to decide whether two superpixels are connected to the same group. In experiments, spatial superpixels were grouped only when $a(s_i, s_j) \geq 0.7$. Since the aim of our work is to recognize manipulation actions rather than achieve best superpixel group segmentations, we set the threshold manually by visualising the superpixel group segmentation result.

People can try to use different thresholds as long as the obtained segmentations are reasonable.

4.3 Temporal Superpixel Connection

Spatial superpixel groups are extended temporally in an online manner. When the next video frame appears, we compare the foreground superpixels in the next frame with the foreground superpixels in the current frame to build temporal connections between foreground superpixels.

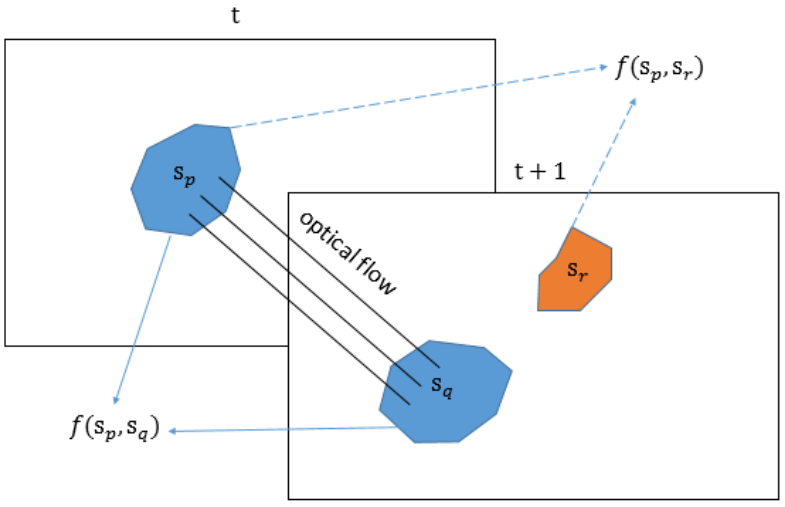


FIGURE 4.2: s_p, s_q, s_r are three foreground superpixels in two consecutive video frames t and $t+1$. $f(s_p, s_r)$ will not be computed since there is no flow between s_p and s_r.

4.3.1 Similarity Measurements

We combine two measurements to compute the similarity between two foreground superpixels in two consecutive frames. The first measurement is the appearance similarity $a(s_p, s_q)$ which is the same colour similarity measurement in spatial connection, where s_p and s_q are two superpixels in two consecutive frames. The second measurement is called flow strength $l(s_p, s_q)$ which is used to establish motion connections. It is defined as:

$$l(s_p, s_q) = \frac{2n(s_p, s_q)}{A(s_p) + A(s_q)} \tag{4.1}$$

where $A(s_p)$ and $A(s_q)$ are the areas of s_p and s_q in pixels, respectively. For each pixel, Farneback's optical flow algorithm [21] is applied to obtain a displacement estimate. The count $n(s_p, s_q)$ is the number of pixels in s_p that, when displaced, are located within superpixel s_q. These two measurements are then combined as the similarity $f(s_p, s_q)$ for s_p and s_q.

Note that we set a constraint here that if there is no flow between the two superpixels, they will not be connected. This constraint avoids calculating the appearance similarity and flow strength on every pair of superpixels in two consecutive frames. As can be seen in Figure 4.2, the similarity between s_p and s_q is computed under the condition that there are flows between these two superpixels. Otherwise, the similarity will not be calculated. $f(s_p, s_q)$ is defined as:

$$f(s_p, s_q) = \begin{cases} \varnothing & if \quad l(s_p, s_q) = \varnothing \\ \gamma a(s_p, s_q) + (1 - \gamma) l(s_p, s_q) & if \quad l(s_p, s_q) \neq \varnothing \end{cases} \tag{4.2}$$

where γ controls the relative weighting of the two measurements. $l(s_p, s_q) = \varnothing$ indicates that there is no flow between s_p and s_q. Since the performance of optical flow algorithm is affected by many factors such as speed of movement and change of illumination, the obtained flow map is usually noisy. Therefore, we set $\gamma = 0.7$ in the experiments to give colour similarity a higher weighting than flow strength. This threshold is also changeable as long as the obtained segmentations are reasonable.

The temporal connection is built frame by frame, which means our algorithm can be used to handle online action recognition as well. When the next video frame appears, temporal connections are built between foreground superpixels in the new frame and the foreground superpixels in the previous frame. It is also an idea that tries to build temporal connections between the new frame and

several previous frames. The temporal connections in multiple frames possibly can provide more information for the transformations of objects in a longer duration. However, in this case, each superpixel needs to compare more superpixels to build temporal connections. The computational cost will be increased significantly as well. We only compute similarities of superpixels in two consecutive video frames, considering the computational efficiency.

Finally, by linking the foreground superpixels spatially and temporally, we can generate spatio-temporal superpixel groups. It should be noted that we call them groups rather than tubes or supervoxels because the structure of our groups is variable. They are not bounding boxes and can include bifurcations and loops.

However, there is still an issue in temporal superpixel connection. Unlike spatial connection mechanism, it is not feasible to simply set a threshold for temporal connection similarity $f(s_p, s_q)$ to cut the graph. Some superpixel groups may have very long durations even with a high similarity threshold. Such groups are more likely to span multiple actions. This makes them not suitable to be utilized for discriminating between action classes. In order to solve this issue, we use a variable threshold for the similarity of temporal superpixel connection.

4.3.2 Variable Threshold for Temporal Connection

The video frame rate in the datasets which we used in the thesis is 30 frames per second. Table 4.1 shows the mean durations of actions in the 50 Salads dataset. In experiments, we find that our spatio-temporal superpixel groups can have very long temporal connections if we just use a fixed similarity threshold for the temporal superpixel connection. These groups cross multiple actions. The crucial problem here is that some of these long-duration groups are corresponding to human motions (e.g., hands and arms) or associated objects. These are important information for distinguishing different manipulation actions. Therefore, we need a method to limit the temporal length of superpixel groups. One solution is to

cut the superpixel groups which have long durations into smaller ones. For instance, we can cut a superpixel group with different fixed temporal length scales (15 frames, 20 frames, 25 frames, etc.). Each scale also can be overlapped with others [116]. As it can be seen in Figure 4.3, the superpixel group G which lasts from video frame t to video frame $t+n$ can be cut into small ones with duration equal to 15 frames ($L_i = 15\ frames$) and 20 frames ($L_j = 20\ frames$). However, these fixed length scales will probably cut connections which have strong temporal similarities. Instead, we develop a variable threshold to cut long groups in this thesis, which considers the strength of temporal similarity.

Actions	Mean duration (frames)
add oil	513
add pepper	233
dress salad	965
mix dressing	343
mix ingredients	354
peel cucumber	1163
cut ingredient	936
place ingredient into bowl	248
serve salad onto plate	1586

TABLE 4.1: Mean durations of actions in the 50 Salads dataset.

Consider two superpixels s_p and s_q. s_p is a superpixel in the previous frame, and s_q is a superpixel in the current frame. Let g denote the group that contains s_p. We use the following condition to connect s_p and s_q:

$$f(s_p, s_q) \geq -\frac{1}{2} \times \frac{c(t(g), \mu, \sigma)}{c(1, \mu, \sigma)} + 1,\ \ s_p \in g \tag{4.3}$$

where $t(g)$ is the duration of superpixel group g so far, and $c()$ is the log cumulative distribution function for the normal distribution with mean μ and variance σ^2.

Specifically, the variable temporal threshold which we require should have the following two characteristics. Firstly, it should have a small value when the duration of the superpixel group is short, since we do not want to stop the temporal connection when it just starts. Secondly, as the duration becomes large, it should

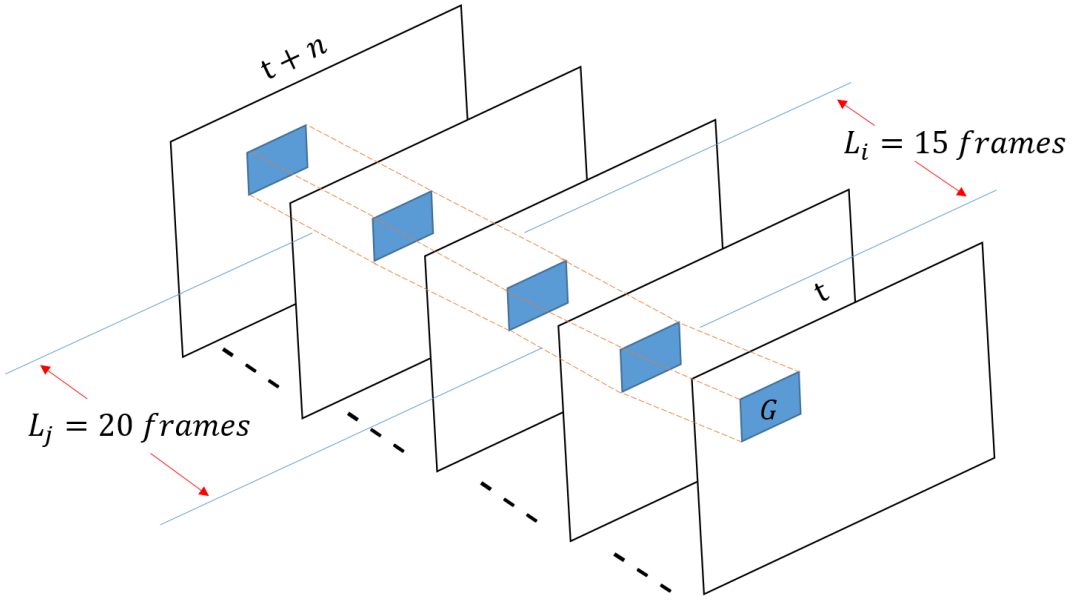


FIGURE 4.3: An example of superpixel group cut using fixed length scales. The two smaller superpixel groups $L_i = 15\ frames$ and $L_j = 20\ frames$ from large group G are allowed to have overlaps with each other.

be able to increase so as to penalise generating long duration superpixel groups. The function $c()$ satisfies these conditions. Since the minimum duration for a superpixel group is one, the minimum log cumulative distribution value is $c(1, \mu, \sigma)$. Equation (4.3) scales $c()$ to a new range [0.5, 1], which means $f(s_p, s_q)$ must not be less than 0.5 when starting to build temporal connections. Therefore, if two superpixels have poor flow and notable differences in appearance at the beginning, they will not be connected. We set $\mu = 50$ and $\sigma = 10$ in the experiments to avoid generating many superpixel groups with very short durations. Figure 4.4 shows the variable temporal threshold with the durations of superpixel groups. We eliminate groups with duration equal to one video frame. It means these groups only have spatial connections so that they are not likely to be helpful to capture the movement information for action recognition.

In summary, our variable temporal threshold method provides a way to capture motions of different temporal scales by generating superpixel groups with different temporal lengths. Comparing with methods using fixed frame lengths, our algorithm considers the strength of temporal similarity when setting the limitation for

superpixel group duration, and lets the system itself decide the temporal length of a superpixel group.

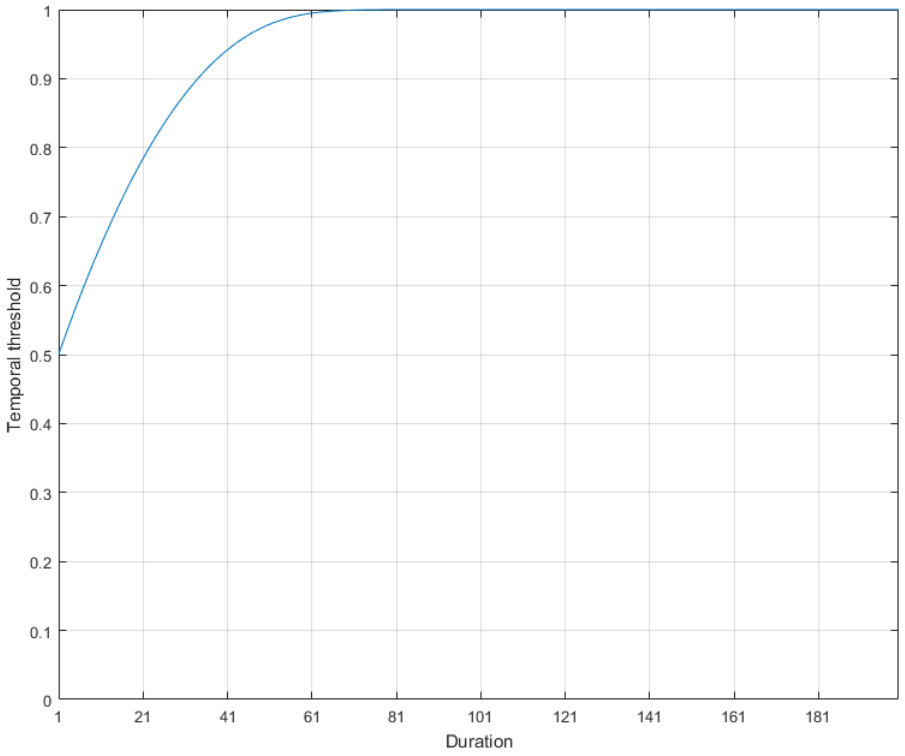


FIGURE 4.4: Temporal threshold with durations of superpixel groups.

By applying our proposed thresholds for spatial superpixel connection and temporal superpixel connection respectively, we are now able to cut the spatio-temporal graph in the videos to generate our spatio-temporal superpixel groups. Examples of groups in the 50 Salads dataset [89] can be seen in Figure 4.5. As shown in the figure, our superpixel groups are able to capture the movement of different sizes of object parts in these fine-grained manipulation actions. Small superpixel groups can locate parts such as the covers of bottles ((a), (b) in Figure 4.5) and a mixing spoon ((e), (i) in Figure 4.5); large superpixel groups can find parts of chopping board ((h) in Figure 4.5) and parts of human hands ((c), (g) in Figure 4.5). Even when related parts become disconnected with each other due to reasons like separated locations, occlusions or noise, our method is able to re-link them to the same group using our grouping mechanism ((c), (h) in Figure 4.5). The durations of these superpixel groups are automatically set by using Equation

(4.3). Groups with mean optical flow magnitude less than one pixel are eliminated since superpixel groups with no significant motion are less likely to be helpful for discriminating manipulation actions.

4.4 Conclusion

In this chapter, we proposed a mid-level representation method which uses superpixels as units to build spatio-temporal superpixel groups. Unlike traditional mid-level representation methods in action recognition, our superpixel groups allow loops and bifurcations. Benefiting from this structure, our proposed groups have the ability to represent actions such as cutting and mixing during manipulations. This provides a way to extract object transformation information in manipulation action recognition. Meanwhile, small superpixel groups are helpful for the system to notice fine-grained changes among different manipulation actions.

When constructing temporal connections between superpixels in two consecutive frames, we proposed a variable threshold approach. It increases the threshold as the duration of superpixel groups becomes large. Long groups will get a high penalty so as to prevent this continuation. Furthermore, comparing with fixed temporal length scale method [116], our variable threshold is linked with the strength of temporal similarity to let the system itself determine the temporal length of a superpixel group.

These spatio-temporal superpixel groups are mainly designed for capturing information from object parts and human body parts. However, the interaction information between different object parts and human body parts in manipulation actions is also a useful cue to distinguish action classes. Therefore, we will introduce a way to create hierarchical superpixel groups in the next chapter.

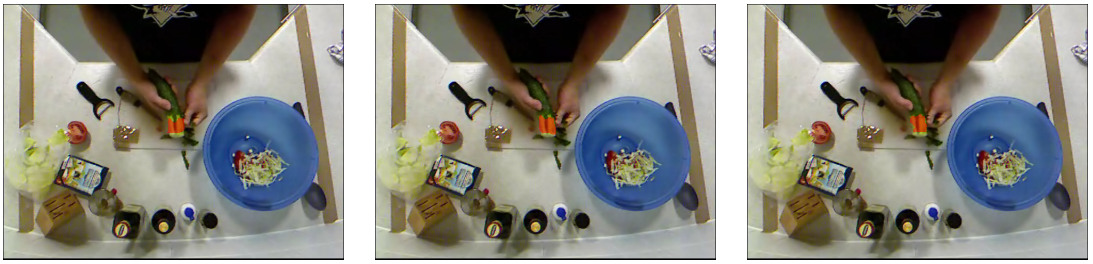
(a) add oil (duration = 10 frames)

(b) add pepper (duration = 19 frames)

(c) dress salad (duration = 31 frames)

(d) mix dressing (duration = 15 frames)

(e) mix ingredients (duration = 8 frames)



(f) peel cucumber (duration = 6 frames)

(g) cut ingredient (duration = 15 frames)

(h) place ingredient into bowl (duration = 21 frames)

(i) serve salad onto plate (duration = 7 frames)

FIGURE 4.5: Examples of superpixel groups (red regions) with their durations for nine action classes in the 50 Salads dataset [89].

Chapter 5

Hierarchical Superpixel Groups

5.1 Motivation and Overview

Our spatio-temporal superpixel groups mainly focus on capturing appearance and motion information for each individual part of objects and part of human body. This could be a strong cue to assist manipulation action recognition. For instance, if parts of tomatoes, parts of the knife and human hands are found in a video sequence, by extracting their appearance and motion features, it can be guessed that the current action in this video sequence is cutting tomatoes. However, apart from using cues from individual object and human body part, interactions between associated objects and human body parts also should bring additional cues for recognizing manipulation actions. We still use the action cutting tomatoes as an example: when cutting tomatoes, the knife is in one of the hands, which means the knife and the hand are attached to each other in terms of locations and they should have similar movements; the knife is also attached to the tomato when cutting in terms of locations and they should have different movements; the other hand usually holds the tomato to fix it on the chopping board, so that they are attached in terms of locations as well and have similar movements. These interactions among different objects and human body parts provide an additional way for analyzing manipulation actions. Therefore, in order to capture interaction

information, we need to build another structure based on our spatio-temporal superpixel groups.

Previous methods in the literature have tried using superpixels or supervoxels to construct hierarchical models for human and object segmentations [31, 37, 66, 107]. Grundmann et al. [31] proposed a hierarchical graph-based algorithm to generate spatio-temporal segmentations of video sequences. They extended Felzenszwalb and Huttenlocher's [19] superpixel segmentation method to 3D version to generate an initial small spatio-temporal regions for video volumes. These small regions were hierarchically merged into large spatio-temporal regions based on the similarities of their local descriptors and then used in video segmentation. Hickson et al. [37] combined depth information, RGB information and temporal information to build segmentations on the 4D space. Specifically, they built a 4D graph from video frames first. Then, the difference of depth between two nodes in the graph was computed to generate the initial video segmentations. After that, the difference of colour between two nodes was used to build another segmentation map. The segmentations using depth information could be used to refine the segmentations using colour information. These segmentations were hierarchically merged into large segmentations like in [31]. Bipartite graph matching was applied to find the segmentation correspondence between two consecutive video sequences. Their method was evaluated to segment objects in public datasets. Lu et al. [66] used hierarchical supervoxel consistency to automatically segment human actions in videos. They applied the work of [105] to form a hierarchy of supervoxels. Markov random field (MRF) was then defined on the hierarchy. The final human action segmentation was obtained by minimizing the energy of the hierarchical MRF.

However, the aims of these methods are to segment each individual object or a human body part. The interaction information between human and objects, and between objects and objects, is not included in their segmentations. In order to analyze interactions among objects and human body parts, based on our spatio-temporal superpixel groups, we propose a simple structure which tries to group different object parts and human body parts as representations for human-object interactions. We call this hierarchical superpixel groups.

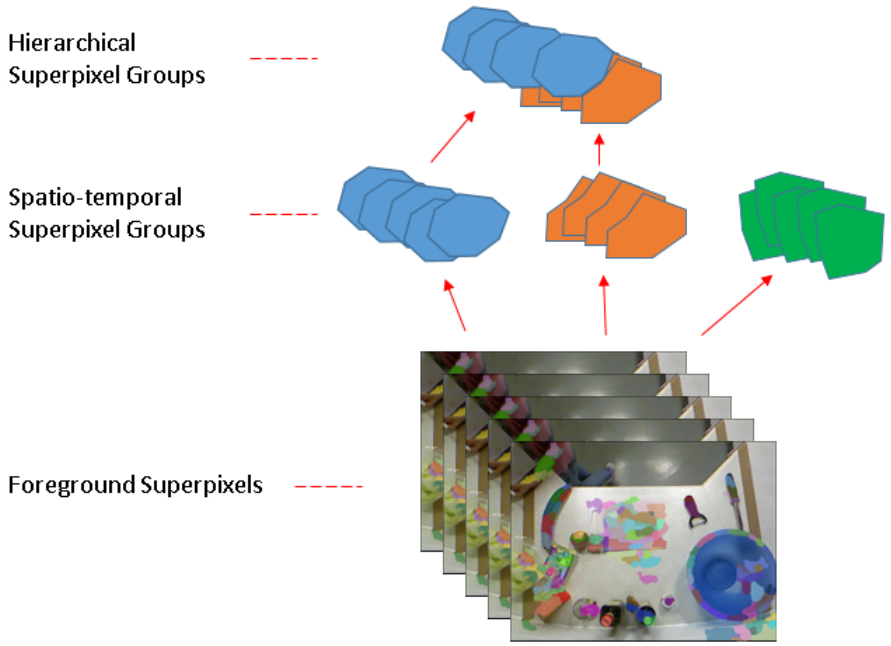


FIGURE 5.1: Foreground superpixels are the basis of this structure. Spatio-temporal superpixel groups are built by connecting foreground superpixels spatially and temporally. The hierarchical superpixel groups are generated by merging spatio-temporal superpixel groups if they have overlaps in the temporal and spatial domains.

In our hierarchical superpixel groups, two spatio-temporal superpixel groups which are overlapped in the temporal domain and spatial domain will be connected into a new group regardless of their appearance similarity and motion similarity. This means that superpixel groups can be combined as a large group as long as they are in contact at some point in the video. The structure from our foreground superpixel segmentation to hierarchical superpixel group segmentation can be seen in Figure 5.1. Foreground superpixels are linked spatially and temporally to create spatio-temporal superpixel groups. Then, these spatio-temporal groups are merged into a large hierarchical superpixel group based on their spatial and temporal locations. For a hierarchical superpixel group, the spatio-temporal groups in it can come from different objects and human body parts.

5.2 Hierarchical Grouping

Connecting foreground superpixels to form our spatio-temporal superpixel groups can be thought as the first level. This level focuses on generating segmentations for each individual object. The hierarchical superpixel groups are built upon the spatio-temporal superpixel groups to form larger groups which contain human-object interaction information.

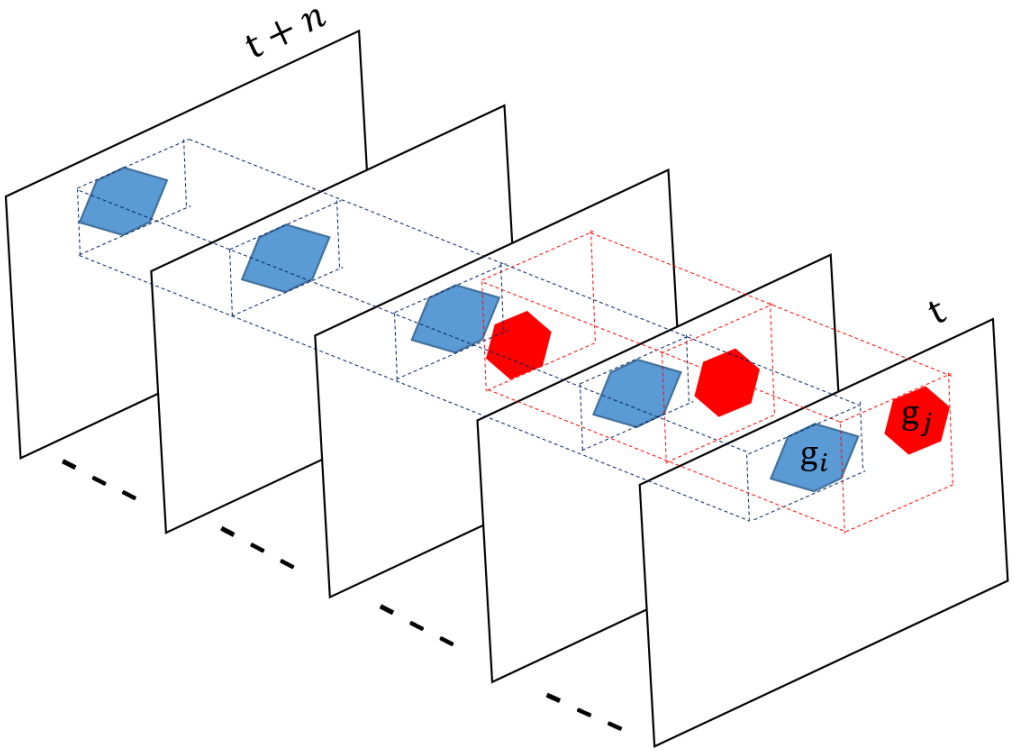


FIGURE 5.2: Example of our hierarchical superpixel grouping. g_i and g_j are two spatio-temporal superpixel groups. The dash lines describe the 3D cuboids for them. g_i and g_j will be merged into a large hierarchical superpixel group since they are overlapped on both temporal domain and spatial domain.

Figure 5.2 illustrates the process. Let's assume that we have two superpixel groups g_i (blue regions) and g_j (red regions) in a video sequence from frame t to frame $t+n$. For each superpixel group, we build a spatio-temporal cuboid which contains the whole superpixel group. We merge two superpixel groups into a large hierarchical superpixel group by two conditions: first, these two groups must have overlaps in the temporal domain; second, the cuboids of the two groups must have overlaps or be attached in the spatial domain. The temporal condition constrains that the

two groups must both exist at a point in time, so that they are likely to be in the same manipulation action. Based on the temporal condition, the spatial condition is used to make sure that the two groups are close to each other so that they are likely to be the associated objects for interactions. The groups g_i and g_j in Figure 5.2 can be merged into a large hierarchical superpixel group since they satisfy both conditions. For each superpixel group, we compare it with other groups which are generated so far. If the temporal and spatial conditions are satisfied, they will be merged into a large hierarchical group with this superpixel group.

Let G denote the set of superpixel groups, T denote the set of temporal indices and S is the set of spatial indices. The process for generating hierarchical superpixel groups H is shown in Algorithm 1. Duplicate hierarchical groups are removed. Superpixel groups which are not connected with other groups are not considered in hierarchical superpixel groups. These hierarchical groups can be merged again to build even larger groups to form a complete hierarchical tree for our superpixel groups. However, this will cost more computational time. We only use the hierarchical groups which are directly generated from superpixel groups in this thesis, for reasons of the efficiency.

Algorithm 1 Hierarchical Superpixel Group Generation

Input: G, T, S
  for *each* $g \in G$ **do**
    $G^{'} \Leftarrow \{g^{'} | g^{'} \in G - g\}$
    for *each* $g^{'} \in G^{'}$ **do**
      if $T(g) \cap T(g^{'}) \neq \varnothing$ **then**
        if $S(g) \cap S(g^{'}) \neq \varnothing$ **then**
          $H(g) \Leftarrow H(g) + g^{'}$
        end if
      end if
    end for
  end for
  $H \Leftarrow RemoveDuplicates(H)$
  $H \Leftarrow RemoveNonHierarchicalGroups(H)$
Output: $H(Hierarchical\ Superpixel\ Groups)$

Figure 5.3 demonstrates some examples for our hierarchical superpixel groups in the 50 Salads dataset [89]. For instance, in action "dress salad" ((c) in Figure

5.3), our hierarchical group (red region) is able to include hands, spoon, food ingredients and the blue bowl together. In action "peel cucumber" ((f) in Figure 5.3), the hierarchical group captures hands, the peeler and the cucumber information in one group. In action "place ingredient into bowl" ((h) in Figure 5.3), the knife, chopping board, food ingredients and hands are merged into one hierarchical superpixel group. Therefore, they demonstrate that our proposed hierarchical superpixel groups can be used to represent interactions happened among different objects and human body parts. These hierarchical superpixel groups can be combined with superpixel groups to recognize manipulation actions.

5.3 Conclusion

We introduced our hierarchical superpixel groups in this chapter. Traditional hierarchical superpixel or supervoxel methods mainly focus on segmenting each individual object or human body part, which lost the information when objects and humans have interactions with each other. Our hierarchical superpixel groups are built upon our spatio-temporal superpixel groups. It can merge different object and human body segmentations into one large hierarchical group. By setting the overlap limitations on temporal and spatial domains, we make the superpixel groups in a hierarchical group be likely to come from associated objects and human body parts during the process of interactions. We can then extract features from these hierarchical groups to represent human-object interactions and use these representations to support manipulation action recognition.

Our hierarchical superpixel groups are also able to be extended to build a completely hierarchical tree for superpixel groups. This provides a chance to analyze more complicated interactions which happen between human and objects, and between objects and objects.

A video can generate lots of superpixel groups and hierarchical superpixel groups based on our proposed algorithms. It may consume much time if all the groups are processed. Instead of that, data mining methods provide a way to find a

small number of discriminative groups for each action. We can then use them to classify different actions. The next chapter will introduce our mining method for superpixel groups.

(a) add oil (duration = 10 frames)

(b) add pepper (duration = 19 frames)

(c) dress salad (duration = 31 frames)

(d) mix dressing (duration = 15 frames)

(e) mix ingredients (duration = 8 frames)

(f) peel cucumber (duration = 6 frames)

(g) cut ingredient (duration = 15 frames)

(h) place ingredient into bowl (duration = 21 frames)

(i) serve salad onto plate (duration = 7 frames)

FIGURE 5.3: Examples of hierarchical superpixel groups (red regions) with their durations for nine action classes in the 50 Salads dataset [89].

Chapter 6

Superpixel Group Mining

6.1 Motivation and Overview

We have introduced how to generate our spatio-temporal superpixel groups and hierarchical superpixel groups in previous chapters. For each video, lots of groups can be generated. However, not all of them are useful for classifying different manipulation actions. Some of the groups can just be noise around objects. Another important reason is that many groups are not representative of their corresponding actions. For instance, the patterns on human clothes may have many superpixel group segmentations in a video. However, they are not helpful to distinguish different actions. In addition, there can be a large numbers of objects in a manipulation video. Some objects possibly are never used or just accidentally touched in actions which are required to be recognized. They will produce many irrelevant superpixel group segmentations as well. If we use all superpixel groups from videos in the system, many irrelevant groups are likely to interfere with the recognition accuracy of manipulation actions. Furthermore, it would increase computational cost.

Instead, each action should have its discriminative superpixel groups. People are able to understand an action by observing discriminative regions. For action "peel cucumber", when a cucumber, a peeler and human hands are found in a video

sequence with their motions, people will have a high confidence to infer the action in this sequence is "peel cucumber". Even without human hands, we are still able to understand the action is "peel cucumber" by observing the movements of the cucumber and the peeler. In this case, superpixel groups which are contained in the cucumber, the peeler and human hands can be considered as discriminative superpixel groups for action "peel cucumber", and the groups that come from the cucumber and the peeler may be more discriminative than human hands. This indicates that we can classify actions by focusing on discriminative superpixel group segmentations for each action.

In the related literature, a mining algorithm is often used to search discriminative information for solving image or video classification problems [43, 110, 116]. Yao and Fei-Fei [110] proposed a method to recognize human and object interactions in static images. The aim of their work is to recognize the interactions between human and objects in a specific manner. For instance, their method is able to distinguish a person playing violin from not playing violin in static images. They mentioned that human pose estimation often could not reliably locate body parts especially when people are occluded or in complicated backgrounds. Therefore, they generated an idea called "grouplets" which capture structured information of an image by using AND/OR structures [12]. As with our superpixel groups, the number of their "grouplets" is quite large. Therefore, they applied a modified Apriori mining algorithm [4] to explore discriminative "grouplets" for each class. These "grouplets" can then be used for classification tasks by integrating into either a discriminative classifier or a generative classifier. Jain et al. [43] utilized the exemplar-SVM (e-SVM) method proposed by Malisiewicz et al. [67] to compute discriminative distance metrics for spatio-temporal patches. A linear combination is then used to combine the appearance consistency score and purity score to rank candidate patches. Top-ranked patches are chosen as discriminative spatio-temporal patches. The result of e-SVM is used to build feature vectors for action classification. Yang et al. [116] proposed an algorithm called "interaction part mining" to recognize manipulation actions in videos. Unlike image patch mining methods in [43, 110], their interaction parts aim to directly find spatio-temporal

volumes which contain motions of human and objects. They applied Juneja et al.'s [46] mining algorithm to find discriminative interaction parts in their recognition system. Juneja et al.'s [46] mining method uses linear discriminant analysis (LDA) [35] to learn part detectors and they proposed a novel idea called entropy-rank to select discriminative parts.

Apart from these traditional mining algorithms, researchers have also developed deep learning mining methods in recent years [60, 61, 118]. For action recognition, Zhu et al. [118] pointed out that many existing action recognition approaches using deep neural networks treat every volume equally. A large amount of irrelevant volumes will lower recognition performance. Therefore, they tried to encourage the deep network to focus on discriminative volumes for action recognition. A set of volume-level binary classifiers are attached after a deep convolutional neural network (CNN) structure in their system. The number of classifiers is the same as the number of categories in the dataset. The input of this network is a bag of volumes. Basically, if the action label of this bag is Y, it is hoped that all volumes in this bag get low responses for classifiers whose corresponding action labels are not equal to Y; for the classifier whose corresponding action label is equal to Y, they intend that the key volumes obtain high responses. They achieved this goal by using Max out [29] and its counterpart called Stochastic Out [118].

In this thesis, we propose a participant-based mining algorithm to assist the system to select discriminative superpixel groups for each action class. We follow the approach in Fernando et al. [25], which considered two crucial criteria for selecting discriminative patterns. One is *discriminativity* which means the selected discriminative patterns in one category should rarely occur in other categories. The other is *representativity* which means that the discriminative patterns should also occur frequently in their corresponding categories. Furthermore, the selected patterns should not just appear frequently in a very few samples since these may be caused by repetitive structures. We follow this two criteria to build our algorithm for superpixel group mining. We represent each superpixel group based on its colour, motion and texture information, and then compute the similarity between groups.

The k nearest neighbors of each participant are selected to calculate a discriminativity score for each superpixel group [41]. For the representativity score of each group, we count the proportion of participants with at least one selected group with the same action label as that group. The mining score for a group is computed by summing its discriminativity and representativity scores [42].

Our manipulation action recognition is achieved by applying a temporal sliding window. The action label of a temporal window is assigned to the middle frame of that window. The temporal window will move frame by frame until the end of a video so that we can obtain action label prediction for each frame in that video. The representation for a temporal window is generated by using Max-N pooling [116] with our discriminative superpixel groups from each action class. Finally, a LibLinear [20] SVM classifier is trained to predict action labels for each temporal window. We will describe details of these steps in this chapter.

6.2 Group Representation and Matching

Our superpixel groups have different sizes in both spatial and temporal domains. In order to compare superpixel groups to explore discriminative superpixel groups for each action class, we need a way to represent each group first and a measurement to judge the similarity between groups. In this thesis, we use three histograms to describe colour, motion and texture information in a superpixel group, respectively. For colour information, each colour channel has 25 bins (we use RGB channels in this work) resulting in a total of 75 bins colour histogram for a superpixel group. This setting is the same as in [93] which they found to work well.

The optical flow orientation histogram weighted by flow magnitudes [10] is applied to represent motion information for a superpixel group. In our experiments, the number of bins for optical flow histogram is 30 which shows a good performance when tuning this parameter in the training set. Figure 6.1 shows an example of optical flow histogram with eight bins. Each flow vector is assigned to one bin

according to its angle. For the same example, only four bins were used in [10] since they expect that the same histogram should be used to indicate whether a person is walking from the right to the left or from the left to the right. However, in the manipulation actions, movements with different directions may refer to different actions. Therefore, the example in Figure 6.1 uses eight bins rather than four bins to represent different directions for optical flow vectors. Each flow vector is weighted by using its magnitude.

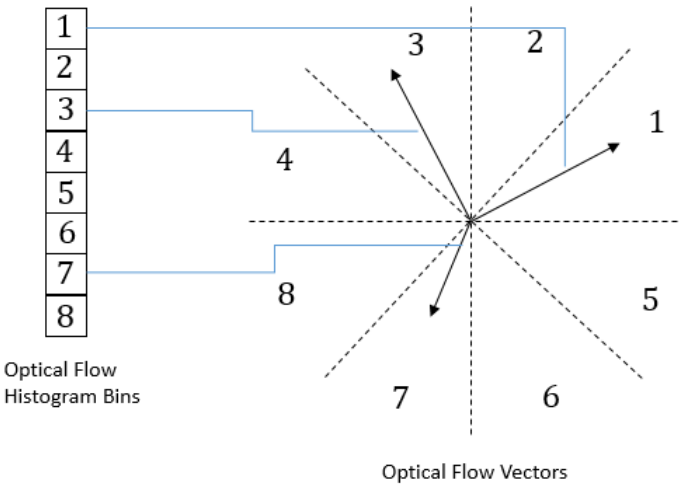


FIGURE 6.1: An example of optical flow histogram with eight bins.

We use a histogram of oriented gradients to extract texture information in a superpixel group. Each pixel gradient vector in the group is allocated to one bin according to its direction and weighted by its magnitude. Each gradient histogram has 30 bins that covers all directions, like in optical flow histogram.

After three histograms are extracted for superpixel groups, we compute the similarity of two superpixel groups by using histogram intersection. Consider two superpixel groups g_i and g_j. Let $a(g_i, g_j)$ denote the intersection of their colour histograms, $m(g_i, g_j)$ denote the intersection of their optical flow histograms and $h(g_i, g_j)$ the intersection of their oriented gradient histograms. The similarity $k(g_i, g_j)$ of g_i and g_j is the weighted sum of the three intersections:

$$k(g_i, g_j) = \beta_1 a(g_i, g_j) + \beta_2 m(g_i, g_j) + \beta_3 h(g_i, g_j) \tag{6.1}$$

where β_1, β_2 and β_3 control the weights of the colour, motion and texture representation similarities, and $\beta_3 = 1 - \beta_1 - \beta_2$. We set the values of β_1 and β_2 in the training and validation process (see Chapter 7).

6.3 Group Mining

After finding the similarity measurement $k(g_i, g_j)$ between two superpixel groups, these groups with different sizes become comparable. We are now able to search for discriminative superpixel groups for each action class. We combine two scores to compute a mining score for each superpixel group: discriminativity score and representativity score.

6.3.1 Discriminativity Score

The meaning of discriminativity is that, for example, if we hope to find discriminative superpixel groups for *action A*, the mined groups should mainly tend to appear in instances of *action A*. These groups should rarely occur in instances of other actions. Specifically, superpixel group g which comes from *action A* is compared with the other groups in the training set. Its k nearest neighbors are selected. Then, the discriminativity score is the proportion of those neighboring groups that are also from *action A*. The idea is that, if a superpixel group is discriminative for *action A*, most of its retrieved k nearest neighbors should also belong to *action A*. Let $y(g)$ denote the action label of g. The discriminativity score d_g is defined as:

$$d_g = \frac{|\{g^{'}|g^{'} \in knn(g) \wedge y(g^{'}) = y(g)\}|}{|\{g^{'}|g^{'} \in knn(g)\}|} \tag{6.2}$$

where $knn(g)$ is the set of k nearest neighbors of group g, $g^{'}$ is the superpixel group in $knn(g)$. Figure 6.2 shows the process of discriminativity score d_g computation for superpixel group g from *action* A using five nearest neighbors.

However, there is an issue that the k nearest neighbors of a group g may all come from the same participant of group g. The groups with high discriminativity scores may be some specific patterns of this participant rather than important patterns for actions. Therefore, we still use Equation (6.2) to calculate discriminativity score for a group g but selecting k nearest neighbors from each participant (except the one who owns group g) rather than selecting from the other groups in the whole training set.

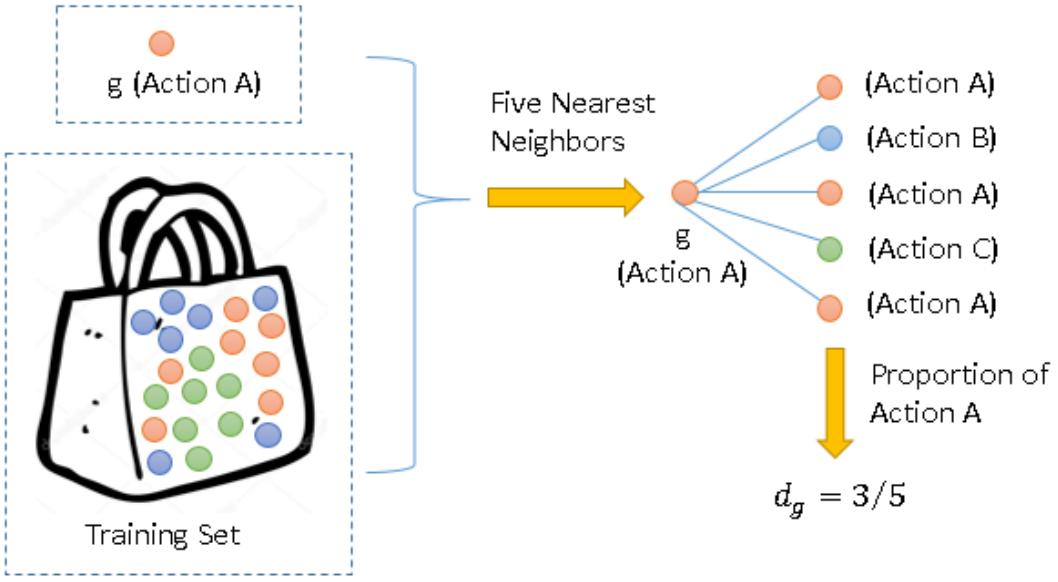


FIGURE 6.2: Discriminativity score d_g of superpixel group g computed with five nearest neighbors. Each circle represents a superpixel group. Note that groups having the same colour only indicates that they are from the same action class. It does not mean that they have similar appearance.

6.3.2 Representativity Score

A discriminativity score is used in most relevant pattern mining methods. However, another important characteristic, the representativity, is often missing. The meaning of representativity is that the discriminative patterns of an action should

occur in most instances of that action. Previous work such as [110, 113] only considered that the discriminative patterns should frequently appear in the corresponding action. However, if an action sequence contains repetitive structures which are not related to this action, their method may select these repetitive structures as discriminative patterns for this action. Such discriminative patterns should not be the best choice for presenting representativity characteristic [25].

We want to select superpixel groups whose patterns are distributed in as many participants who performed the corresponding manipulation actions as possible. Specifically, after selecting k nearest neighbors from each different participant for a superpixel group g in Section 6.3.1, we compute the proportion of participants with at least one neighboring group with the same action label as g. The proportion is our representativity score for group g. We assume that each participant in the training dataset has performed all labelled actions. Let P denote the number of participants in the training set, the total number of retrieved nearest neighbors $knn(g)$ for group g is $N = k \times (P-1)$. Let $Participant(knn(g))$ denote the corresponding participants of $knn(g)$ without duplication and $y(g)$ denote the action label of g. The representativity score r_g of group g is then computed as:

$$r_g = \frac{|Participant(\{x|x \in knn(g) \wedge y(x) = y(g)\})|}{P-1} \tag{6.3}$$

Figure 6.3 presents an example for the computation of representativity score of group g. The action label of g is *Action A* which is performed by *Participant 1*. There are three participants in the training set. We search five nearest neighbors in *Participant 2* and *Participant 3* for g, respectively. The representativity score $r_g = 1/2$ because the number of participants is two excluding *Participant 1*, and only the nearest neighbors from *Participant 2* contain superpixel groups from *Action A*.

The final mining score m_g for superpixel group g is the summation of discriminativity score d_g and representativity score r_g with equal weights since they are both important for mining process based on two different aspects:

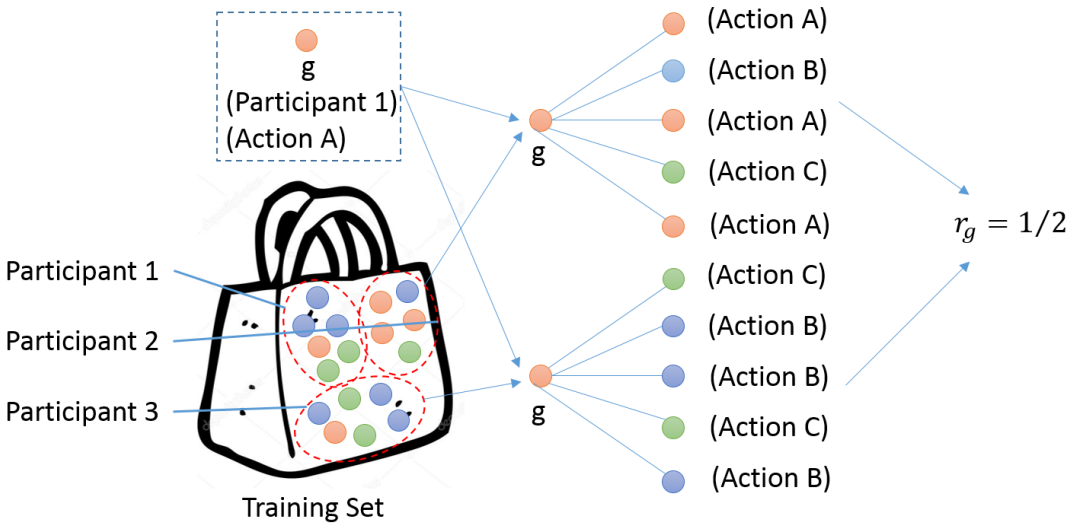


FIGURE 6.3: Representativity score r_g of superpixel group g with five nearest neighbors from each different participant. Each circle represents a superpixel group. Note that the groups having the same colour only indicates that they are from the same action class. It does not mean that they have similar appearance.

$$m_g = d_g + r_g \tag{6.4}$$

For each action class, we select the K highest scoring superpixel groups based on their mining scores as the discriminative superpixel groups for that action class. If the number of action classes is L, for non-hierarchical superpixel groups, the total number of discriminative superpixel groups will be $V = K \times L$. We use the same way to retrieve V discriminative groups for hierarchical superpixel groups as well. These discriminative groups are then used to form a representation for each temporal sliding window.

6.4 Max-N Pooling

Our action recognition uses a temporal sliding window approach to assign an action label to each video frame. It is a fixed length window which covers a sequence of consecutive video frames. Directly recognizing action in each single video frame

is also feasible ([110, 111]). However, temporal motion information is missing in this case. Therefore, we recognize the action in a temporal window rather than a single frame. The predicted action label of the temporal window is treated as the action label of the middle frame in this temporal window. The temporal window slides through the whole video frame by frame to obtain an action label for each video frame. A superpixel group is included in a temporal window only if its start frame and end frame are both within the window. The problem here is to generate a representation for each temporal window so that we can then train a multi-class action classifier based on these representations.

We utilize Max-N pooling to build temporal window representations, which shows a better performance than general max pooling [116]. Specifically, let $G = \{g_1, g_2, ..., g_V\}$ denote the entire set of discriminative superpixel groups for all action classes. Normal max pooling compares each discriminative group in G with all superpixel groups in a temporal window. For each discriminative group, the maximum response of the superpixel groups in this window is stored. These responses are concatenated to form a feature vector as the representation for this temporal window. We use Equation (6.1) to compute the similarity of discriminative groups and the superpixel groups in the temporal window. Mathematically, given groups $X = \{x_1, x_2, ...\}$ in the temporal window, the general max pooling process can be described as:

$$p(g_i) = \max_{x_j \in X} k(g_i, x_j), g_i \in G \tag{6.5}$$

where $p(g_i)$ is the maximum response for group g_i, $k(g_i, x_j)$ can be found in Equation (6.1). The $p(g_i)$ computed from all groups in G are concatenated to form a V-dimensional feature vector which is the representation of this temporal window. Figure 6.4 visually describes the computation of max pooling in our system. Assuming we have three action classes (*blue, green, orange* circles in Figure 6.4) in the training set, we select three discriminative superpixel groups for each action class. Assuming there is a temporal window which contains eight superpixel groups. Each discriminative group is compared with all superpixel groups in the

temporal window to produce a 9×8 matrix of similarities. The maximum value of each row is extracted and then concatenated to form a feature vector for the temporal window. If the temporal window comes from action *blue*, we can expect that the maximum responses from three discriminative groups of action *blue* should be higher than other discriminative groups.

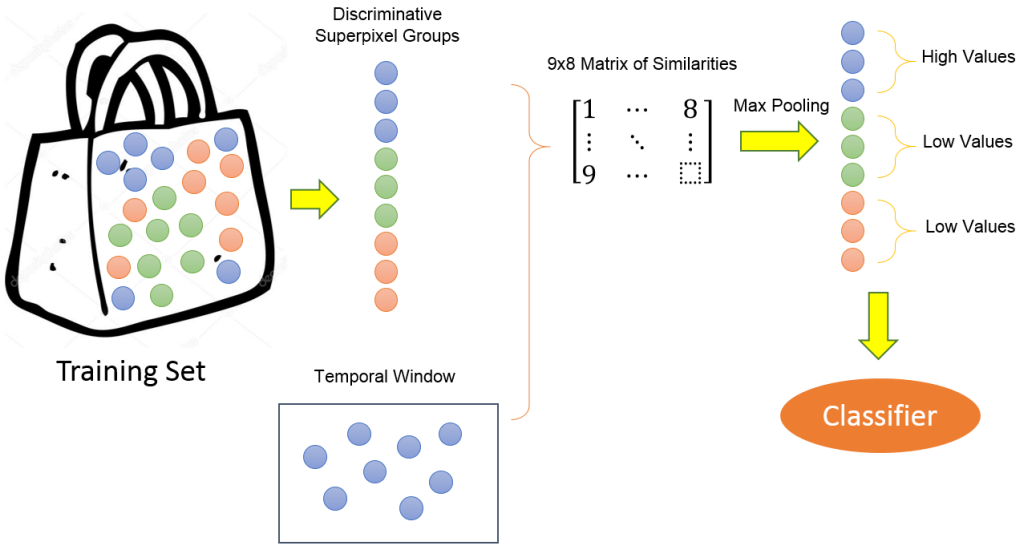


FIGURE 6.4: An example of max pooling for our superpixel groups. Each circle represents a superpixel group. Note that the groups with same colour only indicate that they are from the same action class. It does not mean that they have similar appearance.

A shortcoming of max pooling is that it only selects the highest response for each discriminative group. It may ignore important responses from other superpixel groups in videos. These groups may not return the maximum response for a selected discriminative superpixel group, but they still can be useful to distinguish different actions. The similarities of these groups and discriminative groups also provide information for action classification. Therefore, Zhou et al. [116] proposed using Max-N pooling, which selects the top-N responses for each discriminative superpixel group:

$$\mathbf{p}(g_i) = \underset{x_j \in X}{MaxN}\, k(g_i, x_j), g_i \in G \tag{6.6}$$

where $\mathbf{p}(g_i)$ is an N-dimensional vector and $MaxN$ returns the highest N values. $\mathbf{p}(g_i)$ from all groups in G are concatenated to generate a $(V \times N)$-dimensional feature vector. This feature vector is the representation of this temporal window. We set $N = 2$ following the suggestion in [116].

In the experiments we apply Max-N Pooling on our superpixel groups and hierarchical superpixel groups, respectively. Therefore, each temporal window will obtain one feature vector from superpixel groups and one feature vector from hierarchical superpixel groups. We concatenate these two feature vectors as the final representation for a temporal window.

Figure 6.5 shows some examples of the retrieved discriminative superpixel groups for nine actions in the 50 Salads dataset [89]. Red regions are superpixel group segmentations. As can be seen, some regions present the parts of objects being manipulated and others correspond to human body parts in motion. For instance, the action *add oil* can be represented by combining groups of oil bottle and groups of human hand which interacts with the bottle; focusing on the dressing in the glass with mixing motions can intuitively distinguish the action *mix dressing*; the occurrence of chopping board parts, food ingredients and knife with their motions may infer the current action is *place ingredient into bowl*. The discriminative superpixel groups in actions like *mix ingredients* (e.g., fist image in (e)) also demonstrate that our groups are able to capture object transformation information.

Figure 6.6 presents some examples of discriminative hierarchical superpixel groups for nine actions in the 50 Salads dataset [89]. It shows that the large hierarchical groups indeed provide additional discriminative information to assist manipulation action recognition. For example, it captures the whole bottle of oil rather than a small part of the bottle in the first image of action *add oil*, which has more information to distinguish the action. Meanwhile, discriminative hierarchical groups can include hands and associated objects. It should be noted that our superpixel group and hierarchical superpixel group segmentations are all built without using any pre-trained object detectors.

6.5 Classifier and Post-Processing

6.5.1 Classifier

We use Support Vector Machines (SVM) to classify actions as in many action recognition works (e.g., [11, 52, 100, 101, 116]). Radial basis function (RBF) kernels are often used as the first choice in SVM [39]. This kernel is able to map samples into a high dimensional space in a non-linear way. However, linear SVM is more efficient in processing large data sets and high dimensional data [20]. It only needs to search the penalty parameter C, while RBF SVM requires a grid search for both penalty parameter C and kernel parameter γ. Linear SVM is often used as the first choice in action recognition (e.g., [11, 116]). Therefore, we also apply linear SVM [20] as the classifier to perform manipulation action recognition in this work. We use a one-vs-rest strategy for multi-class classification. The action class with highest score is selected as the predicted class label.

6.5.2 Post-Processing

The classification process in our method is performed window by window, rather than on a sequence of windows. The predicted action label of the current window is not affected by the action labels of previous and latter windows. It causes the distribution of final classified action labels can be scattered. The incorrect predictions of a few frames may appear in a manipulation action.

A simple sliding window with fixed duration is applied on the predicted action labels of video frames to smooth the final classification result. The smoothed action label of the central frame in a temporal window is the action class which obtains the majority vote of all frames within the window.

6.6 Conclusion

We introduced a mining method in this chapter to retrieve discriminative superpixel groups for each action class. It considers two crucial criteria: discriminativity and representativity. For representativity, we select superpixel groups which are distributed in as many participants who performed the corresponding manipulation actions as possible to reduce the influence from repetitive structures in videos. The qualitative examples show that our mined superpixel groups can focus on important parts of objects and human body within manipulation actions. The combination of these discriminative groups provides a strong cue to distinguish different manipulation actions.

A temporal sliding window is used for our manipulation action recognition. We adopted Max-N pooling to generate a representation for each temporal window based on discriminative superpixel groups. We train multi-class linear SVM classifiers for action recognition and use a temporal window with fixed duration to smooth the final predicted action labels.

The description of our manipulation action recognition algorithm have been completed in this chapter. We will evaluate the proposed method on two public datasets and discuss its advantages and disadvantages in the next chapter.



5 10 4

(a) add oil



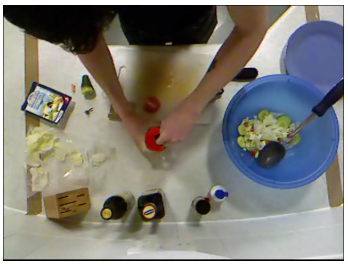
9 9 14

(b) add pepper



6 24 8

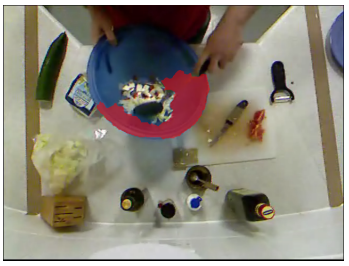
(c) dress salad



11 3 4

(d) mix dressing





21 13 21

(e) mix ingredients

9 14 14

(f) peel cucumber

9 3 7

(g) cut ingredient

12 30 5

(h) place ingredient into bowl

24 31 15

(i) serve salad onto plate

FIGURE 6.5: Examples of discriminative superpixel groups (red regions) for nine action classes in the 50 Salads dataset [89]. Numbers are durations of groups in frames.



13 21 38

(a) add oil



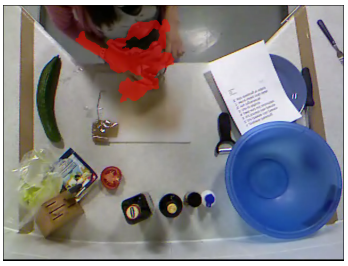
9 16 39

(b) add pepper



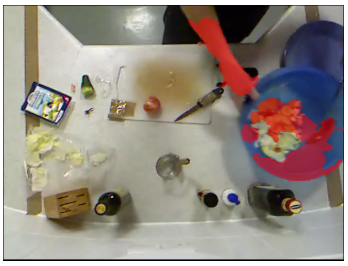
16 24 25

(c) dress salad



23 19 6

(d) mix dressing


23 33 26

(e) mix ingredients

28 17 15

(f) peel cucumber

16 26 29

(g) cut ingredient

45 31 32

(h) place ingredient into bowl

41 35 31

(i) serve salad onto plate

FIGURE 6.6: Examples of discriminative hierarchical superpixel groups (red regions) for nine action classes in the 50 Salads dataset [89]. Numbers are durations of groups in frames.

Chapter 7

Evaluation

We use two public datasets, 50 Salads and Actions for Cooking Eggs (ACE), to evaluate the performance of our proposed method. The details of these two datasets will be introduced first. We discussed the effect of our participant-based mining algorithm compared with mining method which only considers discriminativity for selecting discriminative superpixel groups. The performance of hierarchical superpixel groups is also analyzed in this chapter. In the Actions for Cooking Eggs dataset, we will show that the proposed algorithm can also be used to deal with static scenes. In order to demonstrate the ability of our proposed method, we compare it with several previous methods on both datasets. Our approach achieves the best result on the 50 Salads dataset in terms of frame-wise accuracy and it is also comparable with the methods using object detection and human skin detection in the contest of Actions for Cooking Eggs dataset.

7.1 Datasets

As a result of the increasing attention to recognition of manipulation action which contain challenging fine grained motions, several related datasets are available for researchers. These include TUM Kitchen [92], Georgia Tech Egocentric Activities (GTEA) [22], MPII Cooking [80], Actions for Cooking Eggs (ACE) [85], 50 Salads

[89], Manipulation Action Consequence [108], Manipulation Action (ManiAc) [6] and so forth. Most of these datasets are related to daily living activities, like cooking actions in the kitchen. In this work, we evaluate our proposed algorithm on two datasets: 50 Salads and Actions for Cooking Eggs. 50 Salads is a relatively large dataset which contains over 4 hours of video data. We chose this dataset since it has detailed annotations for actions and has often been used in recent work [24, 54, 55, 115] with different popular methods (e.g., dense trajectories and deep learning) for manipulation action recognition. The Actions for Cooking Eggs dataset is smaller. They both used a fixed Kinect camera to capture manipulation actions. Unlike the datasets in [6, 108] which are against a clean background, 50 Salads dataset and Actions for Cooking Eggs dataset capture more realistic manipulation actions in daily living. Our work is mainly focusing on the 50 Salads dataset; the performance on the Actions for Cooking Eggs dataset suggests a good potential for generalization.

7.1.1 50 Salads Dataset

Stein and McKenna [89] created the cooking dataset 50 Salads. It consists of 50 videos of making mixed salads. Colour and depth images were recorded at 30 frames per second. Figure 7.1 shows the kitchen scene in this dataset. A Kinect camera from a top-down view was installed to capture human actions. Stein and McKenna [89] used a multi-modal approach in the original paper to recognize actions. Therefore, some tools have accelerometers embedded. These can be utilized to locate tools and to track them [88]. The accelerometers are embedded so do not change the shape and colour of the tools. In our work, we use only the RGB-D data from the Kinect for training and testing our approach. The resolution of the RGB and depth data is 640×480 pixels.

25 participants are involved in the dataset. They have varied age and cooking experience. Each of them is assigned to make two mixed salads. The order of steps for making a salad is not unique. Stein and McKenna built an action diagram for task order sampling, see Figure 7.2. Numbers in the figure are probabilities of steps

FIGURE 7.1: A video frame example of the 50 Salads dataset.

for making a salad. This is helpful to guide participants and enhance the dataset by considering different possible orders. Each participant is given a different order of tasks each time when he or she makes a mixed salad.

In the dataset, Stein and McKenna provided action annotations with different levels of granularity. People can select the appropriate levels of granularity for their research directions. The one which they used in the action recognition experiment separates all actions into 10 categories: *add pepper, add oil, mix dressing, peel cucumber, cut ingredient, place ingredient into bowl, mix ingredients, serve salad onto plate, dress salad and NULL*, where *NULL* represents all times when one of those 9 actions is not occurring. In order to compare with their performance, we use the same classification setting in this project. The number of frames for each action class can be seen in Table 7.1.

A variety of occlusions between hands and objects or objects and objects happen in the dataset. The shape and topology of objects can be markedly changed during manipulation actions. In addition, not all participants followed the guide of action diagram. For example, after cutting an ingredient into pieces and placing them into a bowl, some participants picked the large pieces from the bowl and cut them again. These situations increase the complexity of recognition.

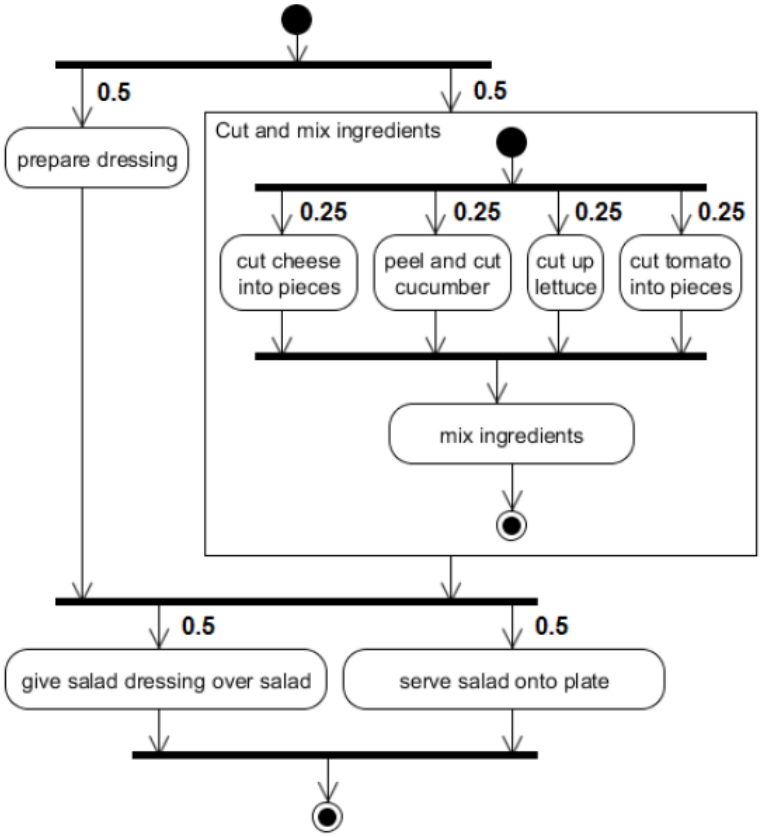


FIGURE 7.2: Action Diagram for the task orderings in [89]. The bar with multiple outgoing arcs represents a choice-node of the diagram. The Numbers are probabilities for choosing each option on the choice-node.

Actions	Frames
add oil	27813
add pepper	12912
mix dressing	19492
peel cucumber	62141
cut ingredient	222193
place ingredient into bowl	58259
mix ingredients	22917
serve salad onto plate	35230
dress salad	22428
NULL	35026
Total	518411

TABLE 7.1: Annotated actions and their corresponding number of frames in the 50 Salads dataset [89].

7.1.2 Actions for Cooking Eggs Dataset

The "Kitchen Scene Context based Gesture Recognition" contest published a dataset called "Actions for Cooking Eggs" (ACE) [85]. It also used a Kinect to capture RGB-D data. Colour and depth images were recorded at 30 frames per second. An example of video frame of this dataset can be seen in Figure 7.3. Table 7.2 demonstrates five different menus used in the dataset. They are popular menus for breakfast except the last one which is a Japanese meal that people usually use to decorate food. Seven participants were recruited. Each of them was required to cook all of the five menus. The resolution of the RGB and depth data is 640×480 pixels.

FIGURE 7.3: An example of video frames in the ACE dataset.

There are eight actions in the dataset: *breaking, mixing, baking, turning, cutting, boiling, seasoning and peeling.* There is also a *NULL* action label which represents none of those eight actions is occurring. Each video frame is assigned to one of the nine action labels by voting from three participants. This dataset also contains many occlusions between human and objects and the occlusions between objects and objects. Furthermore, objects are taken out of view and then brought back by some participants, which increases the difficulty for analyzing the manipulations. Table 7.3 shows the number of frames for each action class in the dataset.

Ham and eggs	Brown some slices of ham, then break eggs on the ham. Season with salt if necessary.
Omelet	Break eggs into a bowl and mix together. Add salt, milk if necessary, and beat again. Pour the egg mixture into the pan and cook for a while until egg mixture is set. Shape the egg with a spatula.
Scrambled egg	Break eggs into a bowl and beat them until they turn a pale yellow colour. Add the ham, salt and/or milk if necessary, and mix them again. Pour in egg mixture. As eggs begin to set, gently pull the eggs across the pan with a spatula until thickened and no visible liquid egg remains.
Boiled egg	Place the raw egg in a saucepan. Run cold water into the saucepan. Boil the water for several minutes. Peel the egg shells.
Kinshi-tamago	Shredded egg crepes, one of Japanese egg recipes. Break eggs into a bowl and mix together. Pour in egg mixture and make a crepe on the pan. With a sharp knife, cut the crepe into thin strips.

TABLE 7.2: The five menus with their introductions in the ACE dataset [85].

Actions	Frames
breaking	6709
mixing	24530
baking	57259
turning	11659
cutting	20193
boiling	16540
seasoning	6990
peeling	13376
NULL	70583
Total	227839

TABLE 7.3: Annotated actions and their corresponding frame numbers in the ACE dataset.

7.2 Experiments for Training Free Parameters

7.2.1 50 Salads Experiments

7.2.1.1 Evaluation Measurements

For the 50 Salads dataset, we split the 50 videos into five folds without overlapping. Each fold has 10 videos which contain both sessions of five participants. Following the experimental setting in [89], we apply five-fold cross-validation to evaluate the performance of proposed methods, which means each fold (10 videos) is tested based on training on the other four folds (40 videos). In order to tune parameters in the algorithm, Stein and McKenna [89] used nested five-fold cross-validation on the training data. Specifically, they split 40 videos in the training data into five folds and then applied five-fold cross-validation again to search parameters. This method is very expensive on computational time, especially when different combinations of parameters need to be trained. Since the dataset is created using a fixed camera, the scene and objects are not changed a lot among different videos. Therefore, we just split 40 videos in the training set into two sets: one set which contains 32 videos from 16 participants is used for training; the other set which contains eight videos from four participants is used for evaluating the performance of parameters. These parameters will be fixed in the five-fold cross-validation on the whole dataset (50 videos).

We use four measurements to evaluate the performance of our approach: mean precision, mean recall, harmonic mean (f-measure) and frame-wise accuracy. Frame-wise accuracy is the proportion of frames that are correctly labelled overall. Given the number of true positive (TP), false positive (FP) and false negative (FN) in the final result, precision, recall and f-measure (F) are defined as follows:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN} \tag{7.1}$$

$$F = \frac{2 \times TP}{2 \times TP + FP + FN}$$

Following the evaluation method in [89], we sum the TP, FP and FN on all testing folds for each action class separately during the five-fold cross-validation. Precision, recall and F are then computed for each class. The mean values of these measurements over all classes are used as performance criteria.

The time complexity for searching the discriminative superpixel groups is $O(n^2)$. In order to create a balanced training set and consider the time complexity, we randomly select 4,000 temporal windows from each action class. 10,000 superpixel groups are then randomly chosen from the 4,000 windows of each class for training. The size of sliding temporal window is 155 frames which is around five seconds as suggested in [90].

7.2.1.2 Parameter Search

We search parameters using the validation dataset (eight videos from four participants). Three important parameters in our approach are β_1 and β_2 (Equation (6.1)), which are used to control the weighting of colour, motion and texture information for computing similarity between superpixel groups; and K, the number of selected discriminative superpixel groups from each action class. The latter determines the dimensionality of feature vectors. Grid search is often used for choosing weight parameters such as β_1 and β_2. We suggest a simplified way to explore the suitable combinations of β_1 and β_2. Firstly, we run three experiments which use colour, optical flow and gradient information to recognize manipulation actions,

β_1 (Colour)	β_2 (Motion)	β_3 (Texture)	Precision (%)	Recall (%)	F (%)	Frame-wise (%)
1	0	0	60	61	59	71
0	1	0	29	32	29	44
0	0	1	38	40	39	51
1/3	1/3	1/3	59	60	58	71
0.6	0.1	0.3	64	65	63	75
0.7	0.1	0.2	**64**	**67**	**64**	**76**

TABLE 7.4: Performance of different combinations of similarity parameters in the 50 Salads dataset.

respectively. Since this is a 10-class classification problem, the accuracy of random guessing should be around 10% for each action class if dataset is balanced. The results in Table 7.4 indicate that all three information are helpful for recognizing actions. Colour features provide the best performance, while optical flow features are worst. Therefore, we assign colour highest weight and optical flow the lowest weight. We tried two further combinations of β_1 and β_2. The best result was achieved when $\beta_1 = 0.7$ and $\beta_2 = 0.1$. We give the result for the three features with equal weights in Table 7.4 as well.

Figure 7.4 shows the influence of the number of selected discriminative superpixel groups, K, from each action class. Initially, the values of precision, recall, F and frame-wise accuracy are increasing with the number of discriminative groups. The performance keeps steady when K is larger than 300. Therefore, we choose $K = 300$ as the fixed parameter for five-fold cross-validation on the whole dataset.

7.2.2 Actions for Cooking Eggs Experiments

7.2.2.1 Evaluation Measurements

This dataset has been separated into training set and testing set as in the contest [85]. The training dataset contains 25 videos which record actions from five participants. Each participant cooked five menus (i.e., five videos). 10 videos made by two participants are used as testing dataset. We extract 10 videos made by two participants in the training dataset as the validation dataset and the rest 15

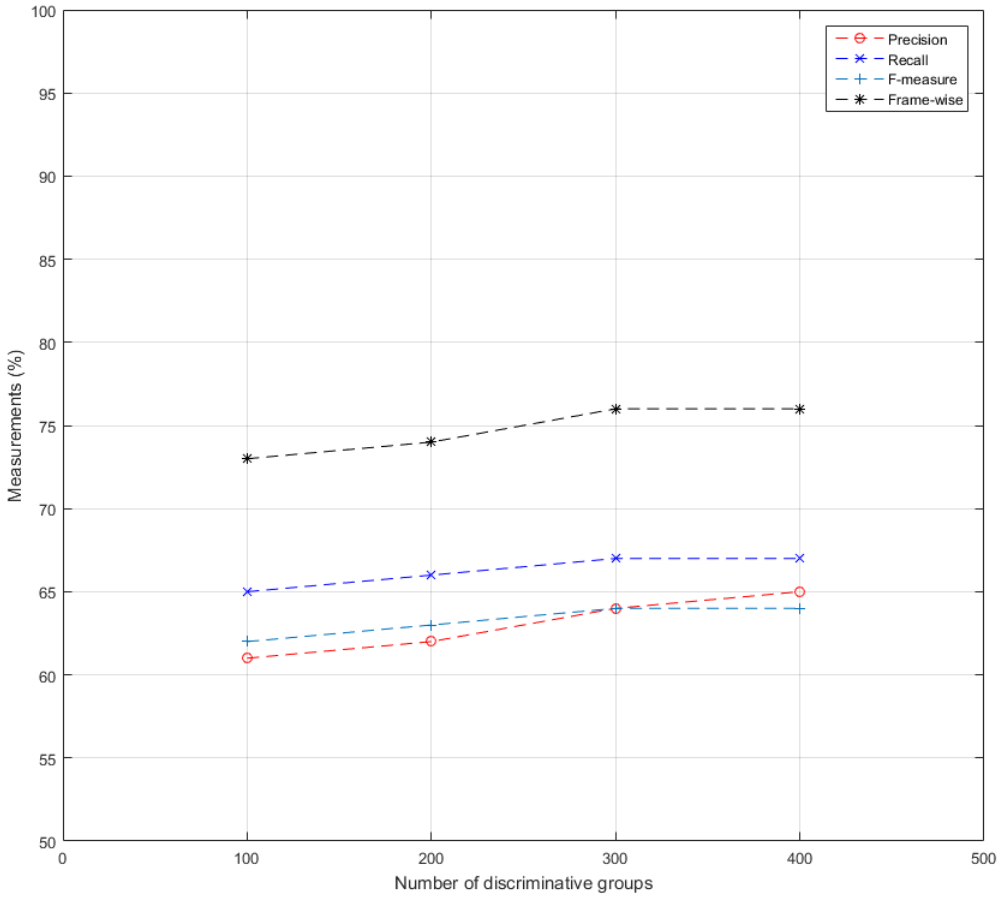


FIGURE 7.4: Precision, recall, f-measure and frame-wise measurements for the action recognition when using different numbers of discriminative superpixel groups for each action class in the 50 Salads dataset.

videos are used as training dataset to tune our parameters. Similar to the 50 Salads experiments, we randomly select 4,000 temporal windows from each action class. 10,000 superpixel groups are then randomly chosen from the 4,000 windows of each class for training. The size of sliding temporal window is the same as for 50 Salads as well (i.e., 155 frames around five seconds).

We use the same measurements in Equation (7.1) of the 50 Salads dataset as the evaluation metrics for the Actions for Cooking Eggs dataset.

β_1 (Colour)	β_2 (Motion)	β_3 (Texture)	Precision (%)	Recall (%)	F (%)	Frame-wise (%)
1	0	0	42	34	36	47
0	1	0	36	36	37	41
0	0	1	42	37	36	43
1/3	1/3	1/3	**54**	**55**	**54**	**56**
0.4	0.3	0.3	53	53	52	55

TABLE 7.5: Performance of different combinations of similarity parameters in the ACE dataset.

7.2.2.2 Parameter Search

We use the same as with the 50 Salads dataset to search the combination of β_1 and β_2, as can be seen in Table 7.5. The performances of colour, motion and texture information were similar. Increasing the weight of colour information was not helpful to improve the recognition result. In the experiment we use equal weights for colour, motion and texture information for the ACE dataset.

Figure 7.5 shows the measurement result when selecting different numbers of discriminative superpixel groups K from each action class in the ACE dataset. The best performance is obtained when $K = 300$. We use this value on the whole training dataset to build a model for testing.

7.2.3 Discussion

The best weighting of colour, motion and texture is different between the 50 Salads and the ACE dataset. A possible reason is that most actions in the 50 Salads dataset have their own associated objects. For instance, "add oil" contains the oil bottle; "peel cucumber" uses the peeler. These associated objects with different colours and textures are a strong signal to distinguish actions. On the contrary, the five cooking menus in ACE dataset are all associated with eggs. The main food ingredients are the same in many actions, which makes it hard to distinguish different actions only by appearance information.

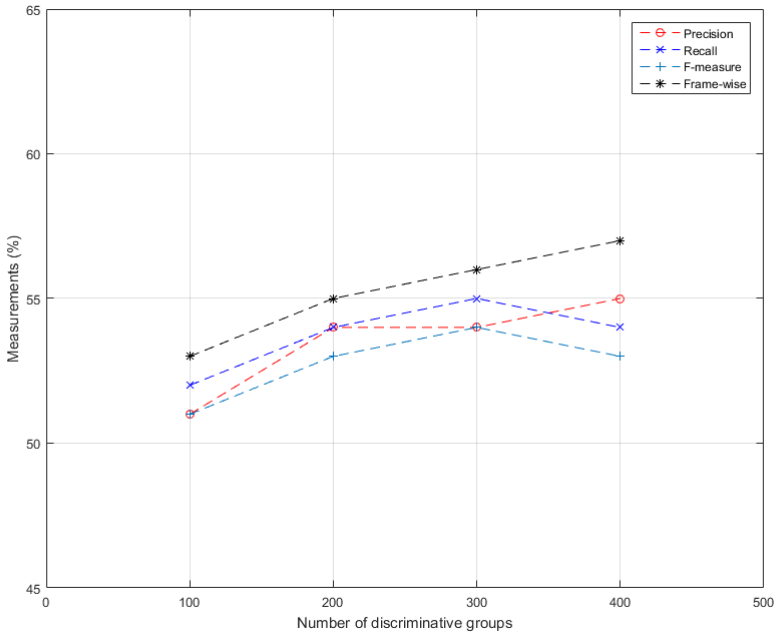


FIGURE 7.5: Precision, recall, f-measure and frame-wise measurements for the action recognition when using different numbers of discriminative superpixel groups for each action class in the ACE dataset.

The best number of discriminative superpixel groups selected from each action class is same on both datasets. For applications on other datasets, people may need more discriminative superpixel groups for each action class if manipulation actions contain many associated objects.

7.3 Superpixel Group Mining Performance

7.3.1 50 Salads Experiments

We compare the performance of the mining method which only uses discriminativity score with our proposed mining method which combines discriminativity score and representativity score. The former method selects k nearest neighbors in the whole training data for each superpixel group to compute its discriminativity score. In our proposed method in Chapter 6, for each superpixel group,

Method	Precision (%)	Recall (%)	F (%)	Frame-wise (%)
discriminativity (19) (k=1)	66	68	67	76
discriminativity+representativity (19) (k=1)	66	68	67	77
discriminativity+representativity (57) (k=3)	65	68	66	76

TABLE 7.6: Performance of mining approaches on the 50 Salads dataset.

we select k nearest neighbors from each different participant. Then, we compute the discriminativity score and representativity score, respectively. The final mining score is the sum of them. We use *discriminativity + representativity* to denote our proposed method. The training set has 20 participants in total. Therefore, *discriminativity + representativity* (19) means that the number of selected nearest neighbors from each different participant is one. *discriminativity + representativity* (57) represents that three nearest neighbors are retrieved from each different participant. The reason for using 57 neighbors is that we hope the discriminative groups can frequently occur in different participants.

Note that the representativity score will be equal to the discriminativity score in terms of *discriminativity + representativity* (19). In this case, it contains both discriminativity and representativity information, since it selects the nearest neighbor from each different participant rather than directly sampling from the whole training set like *discriminativity* method. The results in Table 7.6 show that the three methods have similar performances. We will explore more details in Section 7.3.3.

7.3.2 Actions for Cooking Eggs Experiments

The ACE dataset has fewer participants than the 50 Salads dataset: there are five participants in the training set and two participants in the test set. The *discriminativity + representativity* (4) method means that four nearest neighbors are selected for each superpixel group from four different participants in the

Method	Precision (%)	Recall (%)	F (%)	Frame-wise (%)
discriminativity (4) (k=1)	65	63	63	60
discriminativity+representativity (4) (k=1)	67	63	64	61
discriminativity (12) (k=3)	64	64	63	59
discriminativity+representativity (12) (k=3)	66	64	64	60

TABLE 7.7: Performance of mining approaches on the ACE dataset.

training set (except the participant who the superpixel group is corresponding to). The *discriminativity* (4) method also retrieves four nearest neighbors in the whole training set. *Discriminativity +representativity* (12) selects 12 nearest neighbors in total (three from each participant except the one who has the superpixel group). Since the number of selected nearest neighbors is relatively small for *discriminativity* (4), it may not be able to represent the discriminativity and representativity of the superpixel group. Therefore, we also tried to select 12 nearest neighbors for *discriminativity* method, which is denoted as *discriminativity* (12), and compare it with *discriminativity* + *representativity* (12), as it can been seen in Table 7.7.

7.3.3 Discussion

In order to explore the performance of our mining method, we select the top discriminative superpixel group from action "add pepper" on the 50 Salads dataset by using *discriminativity* (19) and *discriminativity+representativity* (19) methods, respectively, and then analyze the distributions of participants for the k nearest neighbors (*knn*) of the two top discriminative groups.

Table 7.8 is the *knn* distributions of the top discriminative superpixel group for action "add pepper" using *discriminativity* (19) and *discriminativity* + *representativity* (19) methods, respectively. The column "Participants" represents all 20 participants in the training dataset. The column "*knn*" shows the

discriminativity(19)			*discriminativity* + *representativity*(19)		
Participants	*knn*	Same Action Label	Participants	*knn*	Same Action Label
1	0	0	1	1	1
2	1	0	2	1	1
3	0	0	3	1	0
4	0	0	4	1	0
5	0	0	5	1	1
6	0	0	6	1	1
7	0	0	7	1	1
8	1	0	8	1	1
9	1	1	9	1	0
10	0	0	10	1	1
11	0	0	11	1	1
12	0	0	12	1	1
13	0	0	13	1	1
14	0	0	14	1	0
15	13	13	15	0	0
16	3	3	16	1	0
17	0	0	17	1	1
18	0	0	18	1	1
19	0	0	19	1	1
20	0	0	20	1	1

TABLE 7.8: *knn* distributions of two top discriminative superpixel groups for action "add pepper" using *discriminativity* and *discriminativity* + *representativity* mining methods.

distribution of the k nearest neighbors in the 20 participants. "Same Action Label" column is the distribution of the k nearest neighbors which have the same action label with the top discriminative group. In order to demonstrate that this distribution is not a special case, we also show the *knn* distributions of the top discriminative superpixel group for action "dress salad" in Table 7.9.

As can be seen in the two tables, the 19 nearest neighbors for the discriminative group obtained by using *discriminativity* method are clustered in one or two participants. Our *discriminativity* + *representativity* method can split the nearest neighbors into different participants. Only the superpixel group whose nearest neighbors with the same action class occur in different participants can obtain a high mining score. It ensures the representativity is also considered in the mining process.

discriminativity(19)			*discriminativity* + *representativity*(19)		
Participants	*knn*	Same Action Label	Participants	*knn*	Same Action Label
1	0	0	1	1	0
2	0	0	2	1	0
3	0	0	3	1	1
4	0	0	4	1	1
5	0	0	5	1	1
6	0	0	6	1	1
7	0	0	7	1	1
8	0	0	8	1	0
9	0	0	9	1	1
10	19	16	10	1	1
11	0	0	11	1	1
12	0	0	12	1	0
13	0	0	13	1	1
14	0	0	14	1	1
15	0	0	15	1	1
16	0	0	16	1	0
17	0	0	17	1	1
18	0	0	18	1	0
19	0	0	19	0	0
20	0	0	20	1	0

TABLE 7.9: *knn* distributions of two top discriminative superpixel groups for action "dress salad" using *discriminativity* and *discriminativity* + *representativity* mining methods.

Since we eliminate superpixel groups with no significant motion, the remained groups are basically all focus on the human-object interactions. It alleviates the interference from irrelevant patterns. Meanwhile, the manipulation actions in 50 Salads dataset do not have many irrelevant repeated motions. It means even the nearest neighbors of discriminative groups cluster in one participant by using *discriminativity* method, these discriminative groups are still good representations for their corresponding action. These reasons may cause the similar performance between *discriminativity* method and our *discriminativity* + *representativity* method.

To provide more evidences, for the ACE dataset, we also select the top discriminative superpixel groups from action "breaking" based on *discriminativity* and *discriminativity* + *representativity* methods using 12 nearest neighbors. The

discriminativity(12)			*discriminativity* + *representativity*(12)		
Participants	*knn*	Same Action Label	Participants	*knn*	Same Action Label
1	0	0	1	0	0
2	0	0	2	3	3
3	0	0	3	3	2
4	11	11	4	3	3
5	1	0	5	3	2

TABLE 7.10: *knn* distributions of the two top discriminative superpixel groups for action "breaking" using *discriminativity* and *discriminativity* + *representativity* mining methods.

participant distribution of the 12 nearest neighbors is shown in Table 7.10. The *discriminativity*+*representativity* (12) mining method is able to select discriminative superpixel groups whose nearest neighbors with same action label are scattered in different participants, while the nearest neighbors in *discriminativity* method are easy to clustered on few participants.

The improvement of our proposed mining algorithm is not significant compared to the *discriminativity* method. A possible reason is that the number of participants in the ACE dataset is small. Even the nearest neighbors of one superpixel group only occur in two or three participants, the representativity score of this group is high. Another reason, similar to the 50 Salads dataset, is that manipulation actions in the ACE dataset do not have many irrelevant motions.

However, in general, there is no guarantee that irrelevant motions will rarely appear in manipulation actions. The *discriminativity* method will easily capture these irrelevant motions which may only occur in a specific person. In contrast, our proposed mining method which combines both discriminativity score and representativity score is more stable to search discriminative patterns. Section 7.5 will provide more evidence that our participant-based mining method can provide a better performance than general *discriminativity* methods.

Method	Precision (%)	Recall (%)	F (%)	Frame-wise (%)
Superpixel Group	66	68	67	77
Hierarchical Superpixel Group	58	63	60	70
Combined	66	68	67	77

TABLE 7.11: Performance of hierarchical superpixel groups on the 50 Salads dataset.

Method	Precision (%)	Recall (%)	F (%)	Frame-wise (%)
Superpixel Group	66	64	64	60
Hierarchical Superpixel Group	57	56	56	55
Combined	67	62	63	61

TABLE 7.12: Performance of hierarchical superpixel groups on the ACE dataset.

7.4 Hierarchical Superpixel Group Performance

7.4.1 Results on Both Datasets

Hierarchical superpixel groups build upon spatio-temporal superpixel groups. The potential advantage of hierarchical superpixel groups is that they allow one group to contain multiple different object parts and human body parts. We expect that this can achieve better performance.

Table 7.11 and Table 7.12 compare results using non-hierarchical superpixel groups, hierarchical superpixel groups and their combination, on the 50 Salads dataset and the ACE dataset, respectively. The hierarchical superpixel groups does not improve the performance compared with non-hierarchical superpixel groups.

7.4.2 Discussion

The reason for the performance of hierarchical superpixel groups is that the representation used in our hierarchical superpixel groups has shortcomings. We use colour histogram, oriented optical flow histogram and oriented gradient histogram generated from all pixels in a non-hierarchical superpixel group to represent this group. The same strategy is applied on the hierarchical superpixel groups. Since

a non-hierarchical group mainly focuses on one object part movement or human body part movement, even we have two non-hierarchical groups with different sizes, their colour, optical flow and gradient histograms are still the same as long as they are corresponding to the same object part movement or human body part movement. However, the situation is different for hierarchical superpixel groups. Hierarchical superpixel groups can contain multiple object parts or human body parts. We explain this issue in Figure 7.6. Let us assume that there are two hierarchical superpixel groups in the videos (the two rectangles on the top in Figure 7.6) and they are corresponding to the same manipulation action. The size of both groups is 20 pixels. Blue area represents one associated object part or human body part in the hierarchical group. Orange areas represent another associated object part or human body part in the hierarchical group. Since the size of superpixel

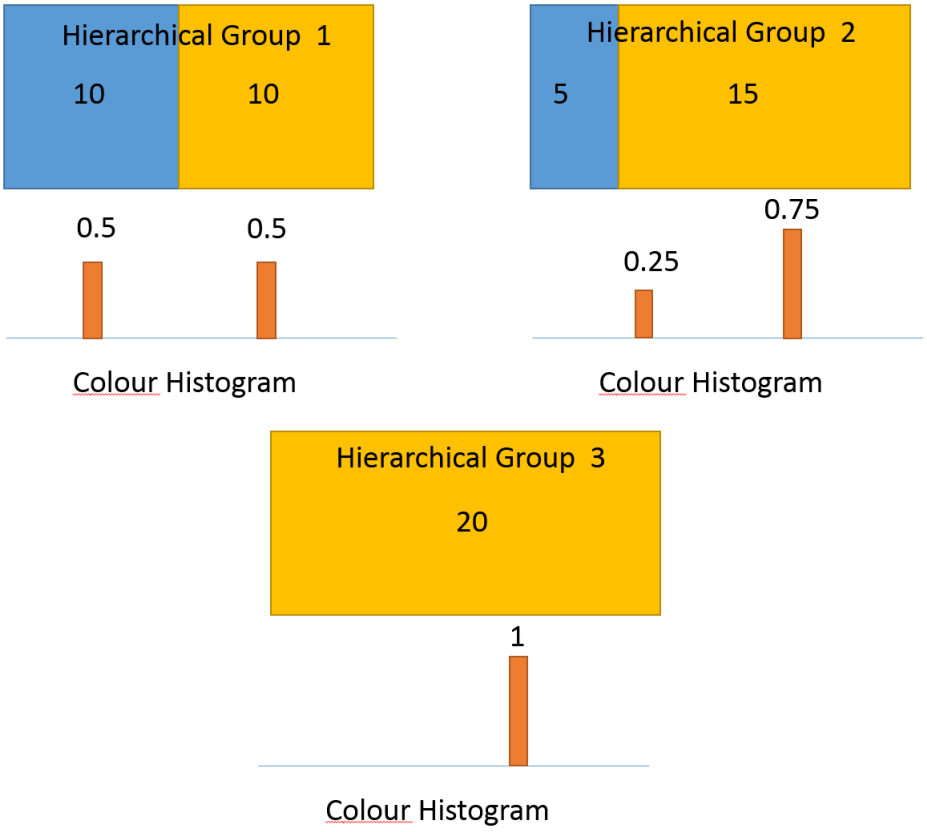


FIGURE 7.6: An example of the shortcoming in hierarchical superpixel groups. Three rectangles represent three hierarchical superpixel groups. They all have 20 pixels. Blue area represents one associated object or human body part. Orange area represents another object or human body part.

segmentations is variable, the sizes of blue part and orange part in different hierarchical groups can be different. Therefore, even though the two hierarchical groups are corresponding to the same action, their colour histograms are not the same. The similarity score of these two hierarchical groups is 0.75 using histogram intersection. If there is another hierarchical group (the bottom rectangle in Figure 7.6) which only captures a large part of the orange object, the similarity between this group and "hierarchical group 2" is also 0.75 using histogram intersection. This causes the problem to find the correct discriminative hierarchical superpixel groups for each action. Therefore, separated representations for different object parts and human body parts in the hierarchical groups are required in the future work.

7.5 Static Scene in ACE

When we discuss recognizing manipulation actions, it is usually assumed that there are human-object interactions in the actions. However, in the ACE dataset, some labelled actions do not have human-object interactions and even no significant motion. Figure 7.7 shows a video frame from action "boiling". The participant

FIGURE 7.7: A video frame example of action "boiling" in the ACE dataset.

Method	Precision (%)	Recall (%)	F (%)	Frame-wise (%)
Superpixel Group without Static	66	64	64	60
Superpixel Group with Static	**70**	**71**	**68**	**72**

TABLE 7.13: Performances of superpixel group with and without static information in the ACE dataset.

is boiling an egg. This static scene can last for several minutes until the egg is ready. The participant just stands there and no interactions happen during the process. Since our method ignores superpixel groups with no significant motion, it is hard for our method to distinguish the actions in the static scene. Therefore, we designed an experiment for the ACE dataset, which includes all static superpixel groups in the training process.

The result is shown in Table 7.13. The superpixel group with static information achieves better performance on all measurements than without using static information. After including static information, the discriminative superpixel groups for action "boiling" are able to capture static patterns. For example, the saucepan is extracted from action "boiling" as a discriminative superpixel group in Figure 7.8, which will not occur in non-static superpixel groups.

FIGURE 7.8: A discriminative superpixel group which captures the saucepan for action "boiling" in the ACE dataset.

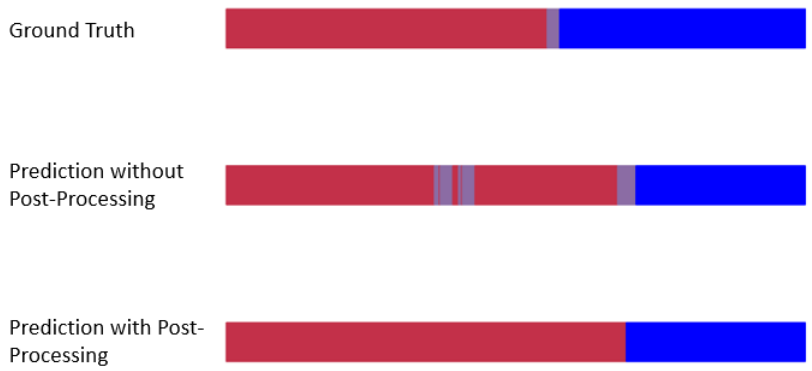


FIGURE 7.9: Performance of post-processing on a video sequence in the 50 Salads dataset.

Our recognition results with static information are obtained based on our proposed participant-based mining algorithm. The total number of superpixel groups increased a lot after adding static information. Many of the static groups are irrelevant for manipulation actions. Therefore, if *discriminativity* method in Section 7.3 is used to search discriminative superpixel groups, the irrelevant patterns will possibly be selected as discriminative groups. The F score is 64% if only including static information and using *discriminativity* as the mining method [42]. This demonstrates an advantage of our participant-based mining algorithm.

7.6 Post-Processing

Figure 7.9 shows the predicted results on a video sequence in the 50 Salads dataset. The prediction without post-processing distributes some incorrect scattered classifications. Therefore, we apply a simple smoothing temporal window approach. The action label of the central frame in the window is the predicted action class which obtains the majority vote of all frames within the window. As it can be seen in the third row of Figure 7.9, these incorrect scattered predictions can be smoothed after post-processing.

Figure 7.10, Figure 7.11 and Figure 7.12 demonstrate the performance when applying smoothing windows with different temporal lengths (i.e., number of frames)

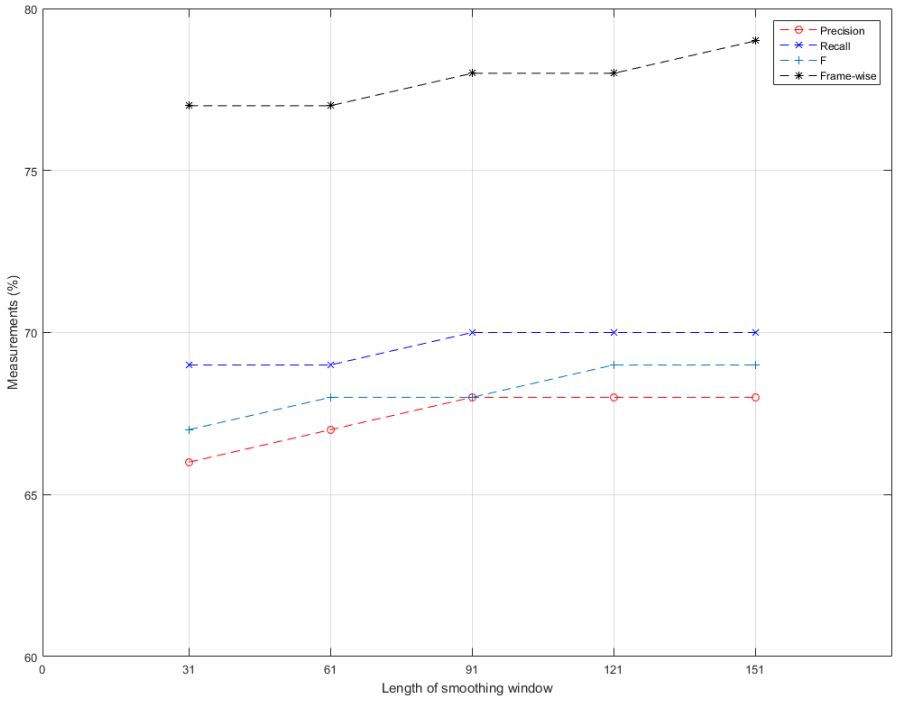


FIGURE 7.10: Precision, recall, f-measure and frame-wise measurements for the action recognition in the 50 Salads dataset with different temporal lengths of smoothing windows.

on the 50 Salads and the ACE dataset. Both datasets achieve better performance after smoothing.

7.7 Comparison with State-Of-The-Art

7.7.1 50 Salads Dataset

For the 50 Salads dataset, we first compare our proposed method with Stein and McKenna's method [90]. They used dense features along tracklets for the manipulation action recognition. We compare our result to the best accuracy obtained in [90] without using accelerometer data. Table 7.14 shows that our method outperform their methods in terms of precision, recall and f-measure. It should be noted that, rather than using dense features in their method, we only have a colour

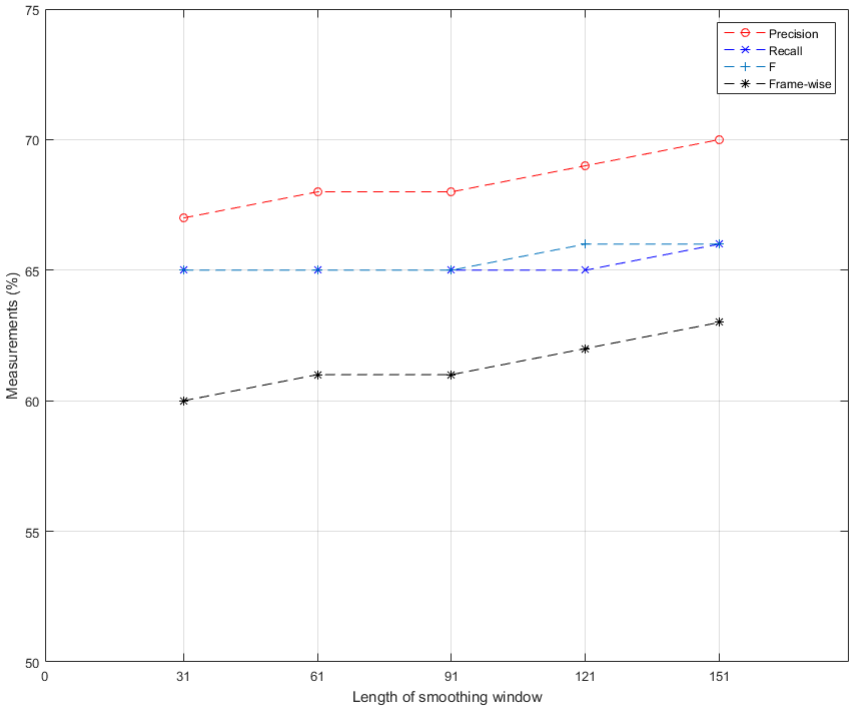


FIGURE 7.11: Precision, recall, f-measure and frame-wise measurements for the action recognition in the ACE dataset with different temporal lengths of smoothing windows. Static information is not used.

Method	Precision (%)	Recall (%)	F (%)
Absolute Tracklets (AT)	42± 2	43± 4	43
HOG	50± 3	49± 3	49
HOF	48± 3	47± 4	47
MBH	54± 5	52± 5	53
AT, HOF, MBH	55± 5	53± 6	54
AT, HOG, HOF, MBH	59± 4	58± 4	58
Ours	**68± 4**	**70± 3**	**69**

TABLE 7.14: Comparison with various feature combinations from [90] on the 50 Salads dataset (± std. dev.). The standard deviation is based on five-fold cross-validation. Therefore, we only show the standard deviation here so as to compare the results in [90].

histogram, an oriented optical flow histogram and an oriented gradient histogram to represent each superpixel group.

We also compare our method with recent published results using deep networks on the 50 Salads dataset. The measurement used is frame-wise accuracy as used

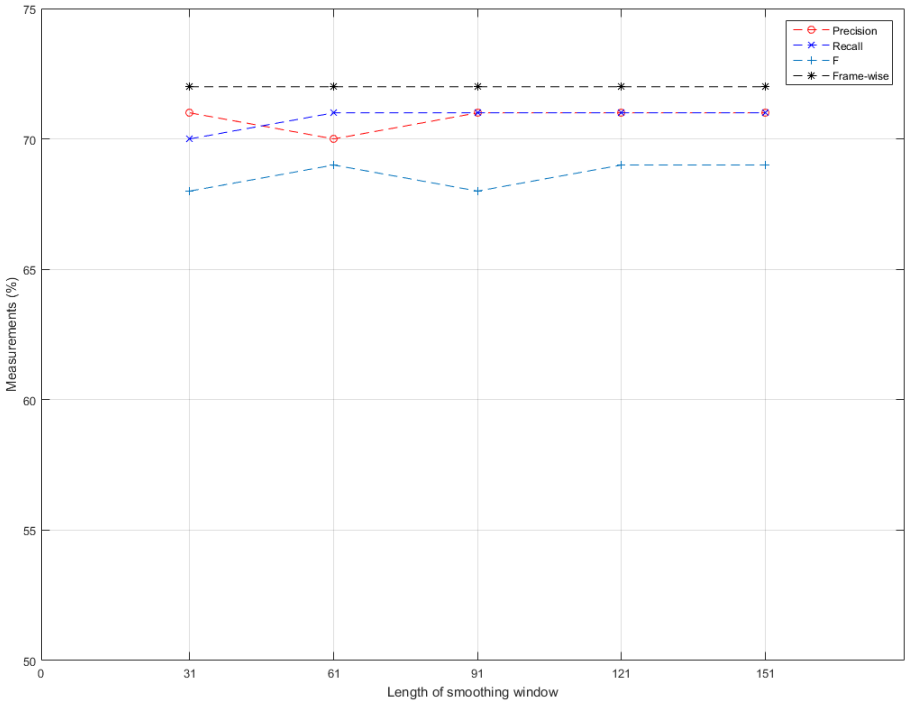


FIGURE 7.12: Precision, recall, f-measure and frame-wise measurements for the action recognition in the ACE dataset with different temporal lengths of smoothing windows. Static information is used.

Method	Frame-wise Accuracy(%)	Look Back (s)	Look Ahead (s)
S-CNN+LSTM [55]	66.3	–	–
S-CNN [55]	66.6	2	0
Bi-LSTM [54]	70.9	–	–
Dilated TCN [54]	71.1	75	75
ST-CNN [55]	71.4	5	5
ST-CNN+Seg [55]	72.0	whole video	
ED-TCN [54]	73.4	26	26
TCED [115]	75.9	–	–
Ours	**78.6**	2.5	2.5

TABLE 7.15: Comparison with deep learning methods on the 50 Salads dataset. "–" means information not clear in that paper.

in [54, 55, 115]. Lea et al. [55] proposed a spatio-temporal CNN model for fine-grained manipulation action recognition. They further developed a temporal convolutional network to improve their previous result [54]. Zhang et al. [115] proposed a novel bilinear pooling operation which can be used in the temporal convolutional encoder-decoder network for the manipulation action recognition.

In Table 7.15, we list the accuracies of deep learning models used in [54, 55, 115]. The S-CNN is the name of spatial CNN model [55]. However, we notice that this S-CNN uses a Motion History Image [98] which actually contains temporal motion information obtained over a two second window. These results indicate that we outperform those CNN based methods in terms of accuracy. Furthermore, the spatio-temporal CNN with Segmental Model (ST-CNN+Seg) and the Encoder-Decoder Temporal Convolutional Networks model (ED-TCN) use long-range temporal information. The ST-CNN+Seg uses the whole video as input for video segmentation. In the experiment [54], the ED-TCN model needs a 53 second temporal window for manipulation action recognition, while we only use five second window in our proposed method. Long temporal windows will delay the manipulation action recognition and are inappropriate for online interactive applications. Zhang et al. [115] also used a temporal convolutional encoder-decoder net with a novel bilinear pooling operation. Fermüller et al. [24] achieved 88.50% frame-wise accuracy on the 50 Salads dataset. However, their method needs manual annotations to build a tracker to track human hands in the videos, which is not fair to be compared with other methods.

Figure 7.13 is the action-class confusion matrix of our method on the 50 Salads dataset. The manipulation actions with relatively low scores are due to the similar actions or same associated objects with other actions. For instance, the "Null" class contains actions such as "add salt" and "add vinegar". Their motions are quite similar with actions "add oil" and "add pepper". Therefore, some frames of actions "add oil" and "add pepper" are misclassified as "Null". Food ingredients are included in both action "dress salad" and action "mix ingredients", which causes the misclassifications between them. Action "mix dressing" has similar motion with action "mix ingredients", and the spoon and glass used in "mix dressing" also appear in action "dress salad". Therefore, some frames of this action are misclassified as "mix ingredients" and "dress salad".

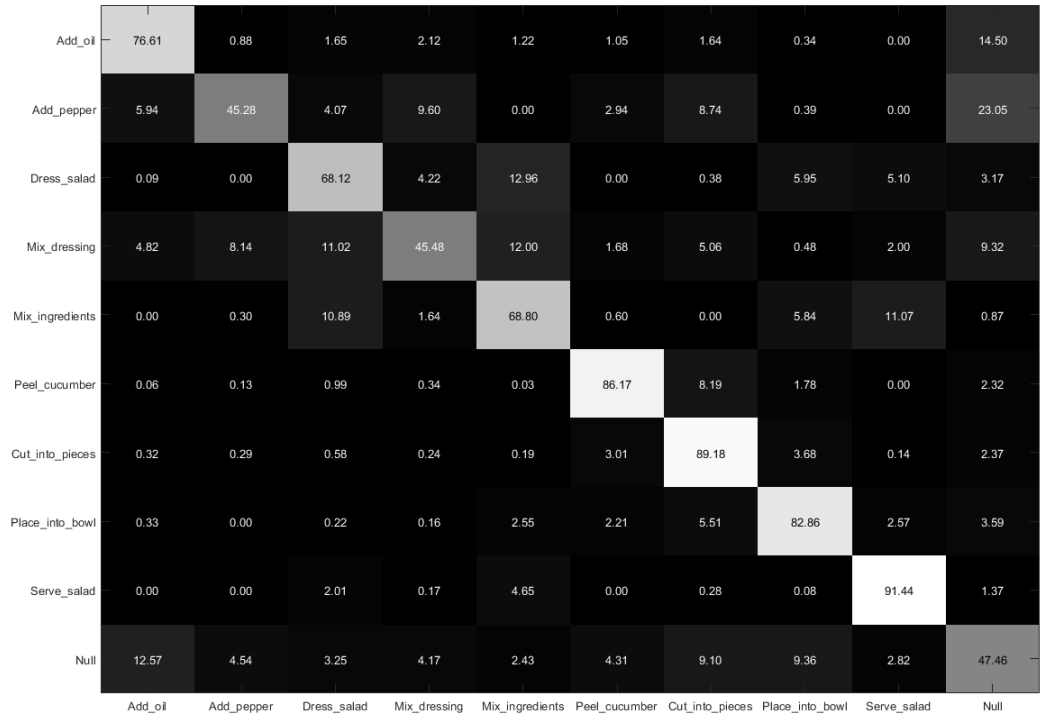


FIGURE 7.13: Confusion matrix for all action classes in the 50 Salads dataset.

7.7.2 Actions for Cooking Eggs (ACE) Dataset

The f-measure is the evaluation method used in the ACE dataset contest. Figure 7.14 presents the recognition results on this contest [85]. "Team-01" and "Team-02" used approaches with object detection and tracking, and human skin detection. "Team-03" and "Team-04" only used the action annotation of each video frame and built models with local features. "Team-05" and "Team-06" achieved the best results. However, the methods utilized in "Team-05" and "Team-06" are heuristic. Their models were built based on the state transition constraints on each cooking menu. They found the menu label for each video first by searching the lengths of some specific states in the video. Then, the manipulation actions were classified based on the duration of each action and strict constraints on the orders of actions in each menu. Their heuristic approaches are quite hard to generalize on other scenarios.

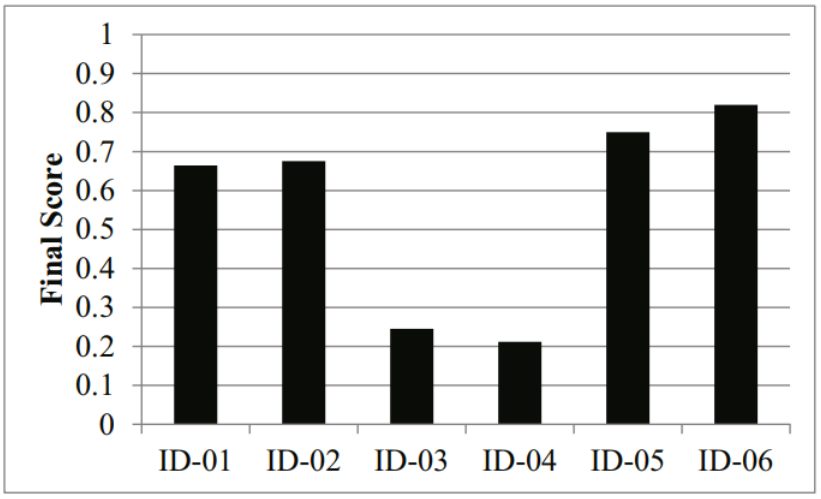


FIGURE 7.14: F-measure score of each team in the contest, taken from [85].

Method	Precision (%)	Recall (%)	F (%)
Context Based SVM	55	54	52
Context Based SVM-HMM	62	63	61
Context Based SVM-HMM (Post-Processing)	68	68	68
Ours	**71**	**71**	**69**

TABLE 7.16: Comparison with context based methods from [9] on the ACE dataset.

The best f-measure score in our proposed methods is 66% without static information and 69% with static information. Our performance is much better than "Team-03" and "Team-04" who also did not use manual annotations for object detection and human detection. Furthermore, our performance is even comparable with the methods which used manual annotations for object and human skin detections in 'Team-01" and "Team-02". Bansal et al. [9] proposed a context based SVM-HMM hybrid model on the ACE dataset by tracking hands and objects. Our method with static information also obtains a better performance than them (see Table 7.16).

The best f-measure score on the ACE dataset is 84% from [71] using deep networks. However, they need manual annotations for object detection to assist to recognize actions.

Figure 7.15 shows the action-class confusion matrices of our proposed methods.

The top one is corresponding to our method without static information and bottom one is with static information. As can be seen in this Figure, using static information alleviates the misclassification situation for actions with many static scenes such as actions "baking" and "boiling". Action "baking" and "turning" often appear in the same region of the video frame and they also used the same objects (e.g., egg and pan). This causes the misclassifications between them. Action "mixing" also has instances that mix food ingredients in the pan, which leads to the misclassifications of some frames in action "turning". The salt container in action "seasoning" is transparent and sometimes is covered by participant hands. Meanwhile, we find the "Null" action labels include the movement when participant hands started to touch the salt container. These cause the incorrect prediction of frames in action "seasoning".

Overall, we believe that our proposed method is a promising solution for recognizing manipulation actions based on the comparisons with other works on the 50 Salads and ACE datasets.

7.8 Conclusion

We evaluated the performance of our proposed methods on the 50 Salads and ACE datasets in this chapter. Both datasets are related to real cooking actions in daily living and have often been used in previous work.

The result from superpixel group mining indicates that our proposed participant-based algorithm can extract discriminative superpixel groups for each manipulation action more effectively by embedding representativity score, compared with methods that only use discriminativity score. Since the actions do not have many repeated irrelevant movements and we remove the superpixel groups with no significant motion, the improvement is not marked on the two datasets. However, when we add static superpixel groups which include many repeated irrelevant patterns in the ACE dataset, our participant-based mining method obtains an apparent improvement.

We explored the static scene in the ACE dataset. It is different from the usual manipulation actions which have human-object interactions. We showed that the result of our method can be improved by adding static information to retrieve discriminative superpixel groups. Our performance on this dataset also shows that our proposed method is able to be applied to other recognition problems even without motion information rather than just for manipulation actions.

The hierarchical superpixel groups did not yield better performance in the experiments compared with non-hierarchical superpixel groups. We listed and discussed the possible reasons.

Our method achieves state-of-the-art result on the 10-class classification problem in the 50 Salads dataset in terms of frame-wise accuracy. It is better than recent deep network methods. Our method achieves much better result than the local feature methods which also only use action label annotation of each video frame on the ACE dataset. Meanwhile, the performance is also comparable with the methods using object detection and human skin detection in the ACE dataset contest.

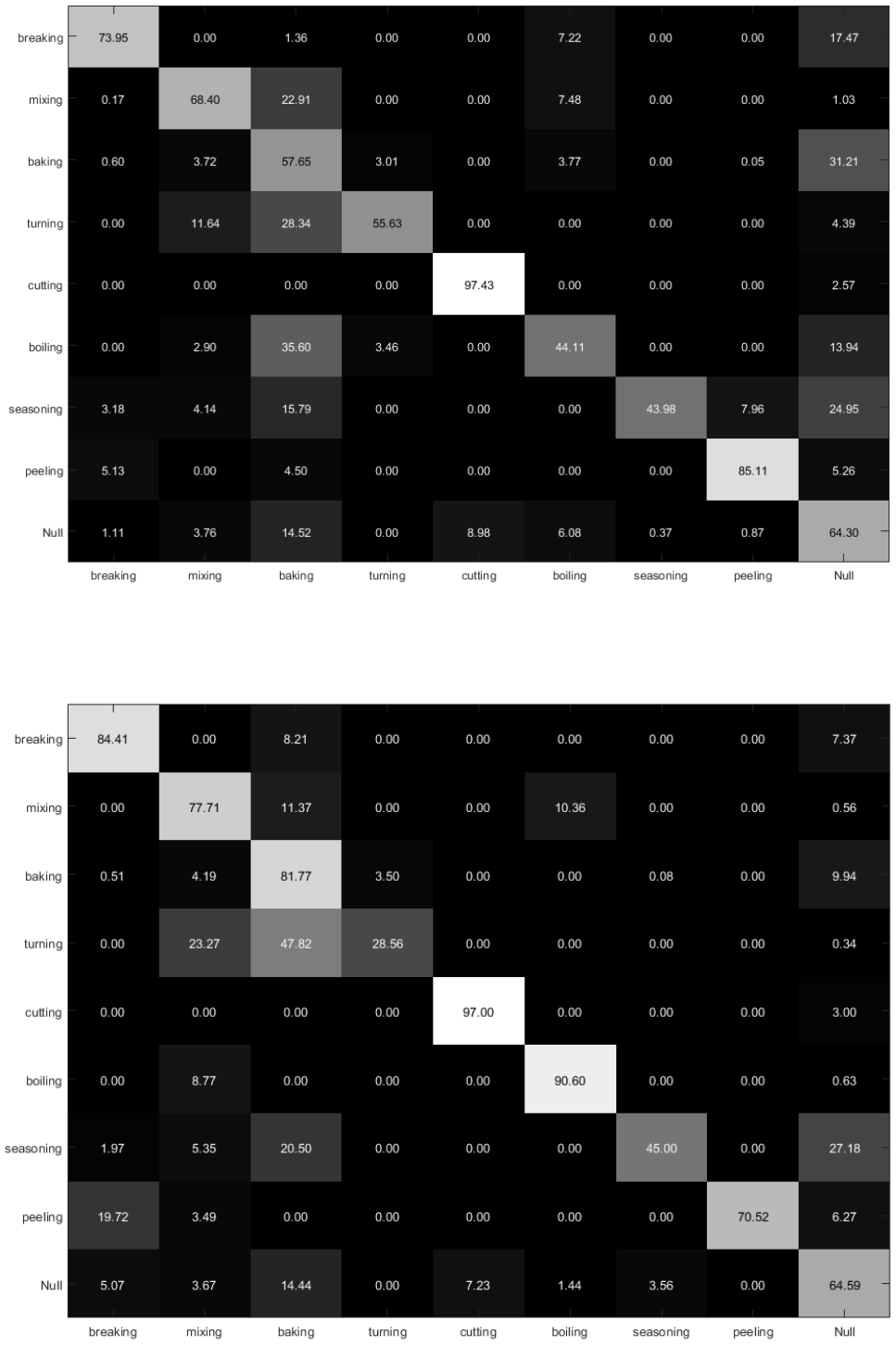


FIGURE 7.15: The top is the action-class confusion matrix of our method without static information. The bottom is for our method with static information.

Chapter 8

Conclusion and Recommendations

8.1 Summary of Contributions

This thesis proposed a novel method for recognizing manipulation actions by generating and mining superpixel groups. The annotations needed in the training process is just the action label for each video frame. Manual annotations for object detection and human body part detection are not required.

A novel mid-level representation approach based on superpixel grouping was introduced. Superpixel groups connect superpixels in each video frame spatially and temporally to construct a spatio-temporal structure. Traditional spatio-temporal tube approaches often assume that the shapes of objects do not change much during actions. However, objects in manipulation actions can experience large topology and shape transformations. For example, in the kitchen, actions like "cutting" and "mixing" will greatly alter the shape and topology of objects. Our proposed superpixel groups can include bifurcations and loops, providing a strategy for representing transformations such as separation and merging in manipulation actions. Meanwhile, our superpixel groups are able to capture small fine-grained motions compared with other methods using large bounding boxes. Furthermore, all of the

superpixel groups are generated in a sequential manner, which makes our method available for online interactive scenarios.

We proposed a participant-based mining algorithm to search discriminative superpixel groups for each action. This mining method uses distributions of participants to compute representativity score for each superpixel group and then combines with discriminativity score to form the final mining score of each superpixel group. Experiment results on two manipulation action datasets consistently show that our participant-based mining algorithm has more effective performance compared with the method which uses only discriminativity score.

Based on the spatio-temporal superpixel groups, we generated a superstructure called hierarchical superpixel groups. These were demonstrated to be able to capture multiple associated objects and human body parts within human-object interactions. Even though the current performance of hierarchical superpixel groups is not ideal, we think that the concept is promising for further investigation.

Two public datasets, 50 Salads and Actions for Cooking Eggs, were used to evaluate our method. We achieved the best performance on the 50 Salads dataset even compared with recently published deep learning approaches. In addition, our model has a directly interpretable representation. Displaying selected discriminative superpixel groups can help developers search for suitable parameters for their applications. Performance was much better than these methods using local features in the Actions for Cooking Eggs contest. The proposed method is also comparable with methods using additional object detection and human skin detection information in the contest. The experiment with static scenes in the Actions for Cooking Eggs dataset indicates that our method is able to be generalized to other recognition tasks even without motion information.

Our method could be used in the absence of depth data. For our general method without using static information, even though the depth is used to reduce the number of superpixels considered for grouping, static regions are subsequently eliminated based on optical flow. For the specific static scene in the Actions for Cooking Eggs, the main purpose of using depth was still to accelerate the

computational process by reducing the number of superpixels. Furthermore, our action recognition is based on colour, texture and optical flow information. We did not extract features from depth maps to represent superpixel groups.

8.2 Recommendations

8.2.1 Training

The amount of training data is important for the recognition performance. Deep learning methods usually need a large training set to generate a good model. In this thesis, the time complexity for searching the discriminative superpixel groups is $O(n_2)$. Considering the efficiency of the experiments, we only selected 10,000 superpixel groups from 4,000 windows for each action in the dataset for training. As can be seen in Table 7.1 and Table 7.3, there are still many more frames that we can use, especially for "Null", that can contain different actions or static scenes. It is possible that randomly selected temporal windows in our approach ignore important features in the unused frames. Therefore, it is worth trying to process all training video frames in future work and to apply methods such as resampling [32] to balance the training data.

8.2.2 Building and Mining Superpixel Groups

Superpixel group creation: In this thesis, we used a fixed number of superpixels to segment each video frame and then connected them to build superpixel groups. The number of superpixels may need to be changed when dealing with different scenarios so as to capture fine-grained motions and small objects. Since scenarios in real environments vary, it may not be a good strategy to assign a specific number of superpixels for each scenario. Multi-scale superpixel segmentation could be a feasible solution to this issue. We can generate superpixels at different scales from fine level to coarse level by segmenting different numbers of

superpixels. Wei et al. [104] proposed a structure which can generate superpixels hierarchically. The number of superpixels in their method is from one to the number of pixels. After obtaining multi-scale superpixels by using these methods, we can connect these superpixels in space and time at different scales to build multi-scale superpixel groups for action recognition. On the one hand, multi-scale superpixel groups solve the segmentation problem in different scenarios; on the other hand, they may also be helpful to improve the recognition performance by adding multi-scale segmentation information when selecting the discriminative superpixel groups. The possible problem with this approach is that it may be very expensive on computational time to generate and process the multi-scale superpixel groups in videos.

Hierarchical superpixel group creation: The hierarchical superpixel groups did not provide a better performance when combined with non-hierarchical groups. If two hierarchical superpixel groups focus on the same human-object interaction, their representations should be the same. However, the sizes of segmentations of associated object parts and human body parts are different among our hierarchical superpixel groups. As it is discussed in Section 7.4.2, even though two hierarchical superpixel groups are corresponding to the same action, their similarity may be not greater than the similarity with a different hierarchical superpixel group. This shortcoming will interfere with the selection of discriminative groups. Therefore, in the future work, the hierarchical superpixel group should have separate representations for each contained object part and human body part.

The hierarchical superpixel group could contain more structure information among associated objects and human body parts. Different manipulation actions may have different geometry information among objects and human body parts. For instance, Yao and Li [110] proposed a method called "grouplet" which encodes the spatial configurations among visual features to distinguish actions like "playing violin" and "not playing violin". We could also embed the related locations of object parts and human body parts in the hierarchical superpixel group representation. For example, the angle between the centres of two superpixel groups in a hierarchical group can be used as a feature for matching hierarchical groups.

This geometry information is not presented in non-hierarchical superpixel groups so that it may help to improve the recognition performance.

Superpixel group mining: Our proposed participant-based mining algorithm is able to force the selected discriminative superpixel groups to appear in different participants, which can help to include representativity into the mining process. However, as we know, the videos of each participant can contain multiple instances of one action. The nearest neighbors which we used to search the discriminative groups in each participant are still possible to gather into one or two instances of one action. Fernando et al. [25] compute the representativity score of a pattern in an image by considering the distribution of this pattern in all images of its corresponding action class. We could use a similar strategy in our approach to require the nearest neighbors selected in each participant to be distributed in different instances of the corresponding action. It may be helpful to locate the more general discriminative patterns for each manipulation action.

8.2.3 Combination with Deep Learning

In order to obtain discriminative superpixel groups of good quality, parameters such as the weights of colour, motion and texture information need to be manually searched in our method. It may take some time to find an appropriate combination. Automatically searching these parameters would be a forward step. One choice is to use deep learning. Deep learning has been widely used in action recognition and recent work also applied it on mining patterns in images [61] and videos [118]. The advantage of deep learning methods is that they are able to build an end-to-end classification system. Parameters in the network will be automatically tuned. Meanwhile, deep learning methods often use information from pre-trained models on other large datasets (e.g., VGG network [86] and ResNet [36]) and then fine-tune the parameters on their own datasets. This inherited information from other datasets is often helpful to obtain a better performance.

One problem for integrating deep learning into our superpixel group method is that the size of input images or video cuboids for a deep neural network is usually required to be fixed. However, our superpixel groups have different sizes in both space and time. The deformable convolutional networks proposed by Dai et al. [14] may be able to solve this issue. They allow additional offsets on the sampling locations in deep networks, so that convolution operations can be applied on deformable shapes, which makes our superpixel groups possible to be connected with deep learning. Lei and Todorovic [57] extended this idea and applied deformable networks to action recognition. These attempts could point to the promising solutions for manipulation action recognition using the proposed superpixel groups.

We used a sliding window approach to recognize manipulation actions in this thesis. The shortcoming of this method is that each temporal window is treated independently. The information in previous temporal windows may be helpful to predict the action label of the current window. Long-short term memory (LSTM) [38] recurrent neural networks [30] could be used in future work to overcome this shortcoming. LSTM includes the information from previous frames and combines with information in current frame to predict the current class label. Meanwhile, LSTM is able to be linked with deep networks so that it can automatically learn the parameters.

8.2.4 Extension to Other Fields

The experiment on the 50 Salads and ACE datasets suggests a good potential for generalization of our proposed method. It can be used in scenarios with different objects and actions.

One limitation of our method and experiments is that the camera is stationary. This is reasonable if the camera is installed to capture a fixed view in the environment. However, this prevents the application of our method to situations which need moving cameras, such as hand-held or body-worn cameras. In addition, even if the camera is installed at a fixed position, it might still rotate so as to obtain

more views. For a moving camera, the problem is to eliminate camera motion. Wang and Schmid [101] removed camera motion by estimating a homography between two consecutive frames. This could also be used in our method to extend our applications, for example, to egocentric video recognition [17, 59, 106].

Another limitation of our method is the work surface detection based on depth information, since it is infeasible to require such a plane to occur in most other recognition tasks. As was mentioned before, the depth information for foreground region segmentation is not necessary in our method since the superpixel groups without significant motion will be eliminated later in the process by using optical flow information. The main reason to use depth information in our work is to reduce the number of segmented superpixels in each video frame so as to accelerate the computation. The ideal approach in future work would be to use all superpixels in all video frames to build superpixel groups and then have discriminative groups for each class retrieved relying on the mining algorithm.

Bibliography

[1] VISINT dataset. *http://www.visint.org/*.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[3] J. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *International Conference on Very Large Data Bases*, pages 487–499, 1994.

[5] E. E. Aksoy, A. Orhan, and F. Worgotter. Semantic decomposition and recognition of long and complex manipulation action sequences. *International Journal of Computer Vision*, 122:84–115, 2017.

[6] E. E. Aksoy, M. Tamosiunaite, and F. Worgotter. Semantic decomposition and recognition of long and complex manipulation action sequences. *Robotics and Autonomous Systems*, 71:118–133, 2015.

[7] M. Andriluka, S. Roth, and B. Schiele. Discriminative appearance models for pictorial structures. *International Journal of Computer Vision*, 99:259–280, 2012.

[8] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence*, 33:898–916, 2010.

[9] S. Bansal, S. Khandelwal, S. Gupta, and D. Goyal. Kitchen activity recognition based on scene context. In *International Conference on Image Processing*, pages 3461–3465, 2013.

[10] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *Computer Vision and Pattern Recognition*, pages 1932–1939, 2009.

[11] F. Chen, N. Sang, C. Gao, and X. Kuang. Discovering distinctive action parts for action recognition. In *International Conference on Image Processing*, pages 1520–1524, 2014.

[12] Y. Chen, L. Zhu, C. Lin, H. Zhang, and A. L. Yuille. Rapid inference on a novel and/or graph for object detection, segmentation and parsing. *Neural Information Processing Systems*, pages 289–296, 2007.

[13] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Computer Vision and Pattern Recognition*, pages 3286–3293, 2014.

[14] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Computer Vision and Pattern Recognition*, pages 764–773, 2017.

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[16] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *European Conference on Computer Vision*, pages 428–441, 2006.

[17] D. Damen, T. Leelasawassuk, O. Haines, A. Calway, and W. Mayol-Cuevas. You-do, I-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. In *British Machine Vision Conference*, 2014.

[18] I. Endres and D. Hoiem. Category independent object proposals. In *European Conference on Computer Vision*, pages 575–588, 2010.

[19] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004.

[20] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: a library for large linear classification. *J. Machine Learning Research*, 9:1871–1874, 2008.

[21] G. Farneback. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, pages 363–370, 2003.

[22] A. Fathi, X. Ren, and J. M. Rehg. Learning to recognize objects in egocentric activities. In *Computer Vision and Pattern Recognition*, pages 3281–3288, 2011.

[23] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence*, 32:1627–1645, 2010.

[24] C. Fermüller, F. Wang, Y. Yang, K. Zampogiannis, Y. Zhang, F. Barranco, and M. Pfeiffer. Prediction of manipulation actions. *International Journal of Computer Vision*, 126:358–374, 2018.

[25] B. Fernando, E. Fromont, and T. Tuytelaars. Mining mid-level features for image classification. *International Journal on Computer Vision*, 108:186–203, 2014.

[26] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[27] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[28] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-time human pose tracking from range data. In *European Conference on Computer Vision*, pages 738–751, 2012.

[29] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning*, 2013.

[30] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[31] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition*, pages 2141–2148, 2010.

[32] H. Guo, Y. Li, J. Shang, M. Gu, Y. Huang, and B. Gong. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[33] Y. Guo, G. Xu, and S. Tsuji. Understanding human motion patterns. In *International Conference on Pattern Recognition*, pages 325–329, 1994.

[34] A. Gupta and L. S. Davis. Objects in action: An approach for combining action understanding and object perception. In *Computer Vision and Pattern Recognition*, 2007.

[35] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classifications. In *European Conference on Computer Vision*, pages 459–475, 2012.

[36] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[37] S. Hickson, S. Birchfield, I. Essa, and H. Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. In *Computer Vision and Pattern Recognition*, pages 344–351, 2014.

[38] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[39] C.-W. Hsu, C.-C. Chang, , and C.-J. Lin. A practical guide to support vector classification. 2003.

[40] C.-D. Huang, C.-Y. Wang, and J.-C. Wang. Human action recognition system for elderly and children care using three stream convnet. In *International Conference on Orange Technologies*, pages 5–9, 2015.

[41] T. Huang and S. McKenna. Sequential recognition of manipulation actions using discriminative superpixel group mining. In *International Conference on Image Processing*, pages 579–583, 2018.

[42] T. Huang and S. McKenna. Superpixel group mining for manipulation action recognition. In *SICSA Workshop on Reasoning, Learning and Explainability*, 2018.

[43] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis. Representing videos using mid-level discriminative patches. In *Computer Vision and Pattern Recognition*, pages 2571–2578, 2013.

[44] M. Jain, J. v. Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek. Action localization with tubelets from motion. In *Computer Vision and Pattern Recognition*, pages 740–747, 2014.

[45] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 14(2):201–211, 1973.

[46] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *Computer Vision and Pattern Recognition*, pages 923–930, 2013.

[47] K. K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24:971–981, 2013.

[48] A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman. Human focused action localization in video. In *Trends and Topics in Computer Vision*, 2012.

[49] H. Kuehne, J. Gall, and T. Serre. An end-to-end generative framework for video segmentation and recognition. In *Winter Conference on Applications of Computer Vision*, pages 1–8, 2016.

[50] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *International Conference on Computer Visionn*, pages 2556–2563, 2011.

[51] T. Lan, Y. Zhu, A. Roshan Zamir, and S. Savarese. Action recognition by hierarchical mid-level action elements. In *International Conference on Computer Vision*, pages 4552–4560, 2015.

[52] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition*, 2008.

[53] C. Lawrence Zitnick and D. Piotr. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405, 2014.

[54] C. Lea, M. Flynn, R. Vidal, A. Reiter, and G. Hager. Temporal convolutional networks for action segmentation and detection. In *Computer Vision and Pattern Recognition*, pages 156–165, 2017.

[55] C. Lea, A. Reiter, R. Vidal, and G. D. Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conference on Computer Vision*, pages 36–52, 2016.

[56] D.-S. Lee. Effective Gaussian mixture learning for video background subtraction. *Pattern Analysis and Machine Intelligence*, 27:827–832, 2005.

[57] P. Lei and S. Todorovic. Temporal deformable residual networks for action segmentation in videos. In *Computer Vision and Pattern Recognition*, pages 6742–6751, 2018.

[58] F.-F. Li and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition*, pages 524–531, 2005.

[59] Y. Li, A. Fathi, and J. M. Rehg. Learning to predict gaze in egocentric video. In *International Conference on Computer Vision*, pages 3216–3223, 2013.

[60] Y. Li, L. Liu, C. Shen, and A. van den Hengel. Mid-level deep pattern mining. In *Computer Vision and Pattern Recognition*, pages 971–980, 2015.

[61] Y. Li, L. Liu, C. Shen, and A. van den Hengel. Mining mid-level visual patterns with deep cnn activations. *International Journal on Computer Vision*, 121:344–364, 2016.

[62] Z. Li, W. Wang, N. Li, and J. Wang. Tube convnets: Better exploiting motion for action recognition. In *International Conference on Image Processing*, pages 3056–3060, 2016.

[63] C. Liu and D. B. Rubin. ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5(1):19–39, 1995.

[64] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *Computer Vision and Pattern Recognition*, 2009.

[65] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition*, pages 2097–2104, 2011.

[66] J. Lu, R. Xu, and J. J. Corso. Human action segmentation with hierarchical supervoxel consistency. In *Computer Vision and Pattern Recognition*, pages 3762–3771, 2015.

[67] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *International Conference on Computer Vision*, 2011.

[68] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *Computer Vision and Pattern Recognition*, pages 2929–2936, 2009.

[69] B. Ni, V. R. Paramathayalan, T. Li, and P. Moulin. Multiple granularity modeling: A coarse-to-fine framework for fine-grained action analysis. *International Journal of Computer Vision*, 120:28–43, 2016.

[70] B. Ni, V. R. Paramathayalan, and P. Moulin. Multiple granularity analysis for fine-grained action detection. In *Computer Vision and Pattern Recognition*, pages 756–763, 2014.

[71] B. Ni, X. Yang, and S. Gao. Progressively parsing interactional objects for fine grained action detection. In *Computer Vision and Pattern Recognition*, pages 1020–1028, 2016.

[72] B. Ni, X. Yang, and S. Gao. Progressively parsing interactional objects for fine grained action detection. In *Computer Vision and Pattern Recognition*, pages 1020–1028, 2016.

[73] J. Niebles, C. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European Conference on Computer Vision*, pages 392–405, 2010.

[74] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *European Conference on Computer Vision*, pages 737–752, 2014.

[75] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *International Conference on Computer Vision*, pages 1817–1824, 2013.

[76] B. Packer, K. Saenko, and D. Koller. A combined pose, object, and feature model for action understanding. In *Computer Vision and Pattern Recognition*, pages 1378–1385, 2012.

[77] A. Prest, V. Ferrari, and C. Schmid. Explicit modeling of human-object interactions in realistic videos. *Pattern Analysis and Machine Intelligence*, 35(4):835–848, 2013.

[78] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *Computer Vision and Pattern Recognition*, pages 3131–3140, 2016.

[79] J. Richter, C. Wiede, E. Dayangac, A. Shahenshah, and G. Hirtz. Activity recognition for elderly care by evaluating proximity to objects and human skeleton data. In *International Conference on Pattern Recognition Applications and Methods*, pages 139–155, 2016.

[80] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *Computer Vision and Pattern Recognition*, pages 1194–1201, 2012.

[81] M. Rohrbach, A. Rohrbach, M. Regneri, S. Amin, M. Andriluka, M. Pinkal, and B. Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. *International Journal of Computer Vision*, 119:346–373, 2016.

[82] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *Computer Vision and Pattern Recognition*, pages 1234–1241, 2012.

[83] S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In *Neural Information Processing Systems*, pages 1185–1192, 2004.

[84] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *International Conference on Pattern Recognition*, pages 32–36, 2004.

[85] A. Shimada, K. Kondo, D. Deguchi, G. Morin, and H. Stern. Kitchen scene context based gesture recognition: a contest in ICPR2012. In *Advances in Depth Image Analysis and Applications*, pages 168–185, 2013.

[86] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[87] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, pages 73–86, 2012.

[88] S. Stein and S. J. McKenna. Accelerometer localization in the view of a stationary camera. In *Computer and Robot Vision*, pages 109–116, 2012.

[89] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 729–738, 2013.

[90] S. Stein and S. J. McKenna. Recognising complex activities with histograms of relative tracklets. *Computer Vision and Image Understanding*, 154:82–93, 2017.

[91] R. Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

[92] M. Tenorth, J. Bandouch, and M. Beetz. The tum kitchen data set of everyday manipulation activities for motion tracking and action recognition. In *International Conference on Computer Vision Workshops*, 2009.

[93] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 2013.

[94] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool. SEEDS: Superpixels extracted via energy-driven sampling. In *European Conference on Computer Vision*, pages 13–26, 2012.

[95] M. Van den Bergh, D. Carton, and L. Van Gool. Depth SEEDS: Recovering incomplete depth data using superpixels. In *IEEE Workshop on Applications of Computer Vision*, pages 363–368, 2013.

[96] M. Van den Berghand, X. Boix, G. Roig, and L. Van Gool. Online video seeds for temporal window objectness. In *International Conference on Computer Vision*, pages 377–384, 2013.

[97] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *CoRR abs/1609.03499*, 2016.

[98] J. W. Davis and A. F. Bobick. The representation and recognition of action using temporal templates. In *Computer Vision and Pattern Recognition*, pages 928–934, 1997.

[99] C. Wang, Y. Wang, and A. L. Yuille. An approach to pose-based action recognition. In *Computer Vision and Pattern Recognition*, 2013.

[100] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79, 2013.

[101] H. Wang and C. Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision*, pages 3551–3558, 2013.

[102] L. Wang, Y. Qiao, and X. Tang. Motionlets: Mid-level 3d parts for human motion recognition. In *Computer Vision and Pattern Recognition*, pages 2674–2681, 2013.

[103] P. Wei, Y. Zhao, N. Zheng, and S.-C. Zhu. Modeling 4d human-object interactions for joint event segmentation, recognition, and object localization. *Pattern Analysis and Machine Intelligence*, 39(6):1165–1179, 2017.

[104] X. Wei, Q. Yang, Y. Gong, M.-H. Yang, and N. Ahuja. Superpixel hierarchy. *Transactions on Image Processing*, 27:4838–4849, 2018.

[105] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *European Conference on Computer Vision*, pages 626–639, 2012.

[106] J. Xu, L. Mukherjee, Y. Li, J. Warner, J. M. Rehg, and V. Singh. Gaze-enabled egocentric video summarization via constrained submodular maximization. In *Computer Vision and Pattern Recognition*, pages 2235–2244, 2015.

[107] Y. Xu, D. Song, and A. Hoogs. An efficient online hierarchical supervoxel segmentation algorithm for time-critical applications. In *British Machine Vision Conference*, 2014.

[108] Y. Yang, C. Fermuller, and Y. Aloimonos. Detection of manipulation action consequences (MAC). In *Computer Vision and Pattern Recognition*, pages 2563–2570, 2013.

[109] Y. Yang, Y. Li, C. Fermuller, and Y. Aloimonos. Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web. In *AAAI Conference on Artificial Intelligence*, 2015.

[110] B. Yao and L. Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. In *Computer Vision and Pattern Recognition*, pages 9–16, 2010.

[111] B. Yao and L. Fei-Fei. Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses. *Pattern Analysis and Machine Intelligence*, 34:1691–1703, 2012.

[112] R. Yi, Y.-J. Liu, and Y.-K. Lai. Content-sensitive supervoxels via uniform tessellations on video manifolds. In *Computer Vision and Pattern Recognition*, pages 646–655, 2018.

[113] J. Yuan, Y. Wu, and M. Yang. Discovery of collocation patterns: from visual words to visual phrases. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[114] S. Zhang, G. Changxin, J. Zhang, C. Feifei, and N. Sang. Discriminative part selection for human action recognition. *Transactions on Multimedia*, 20:769–780, 2018.

[115] Y. Zhang, S. Tang, K. Muandet, C. Jarvers, and H. Neumann. Local temporal bilinear pooling for fine-grained action parsing. In *Computer Vision and Pattern Recognition*, pages 12005–12015, 2019.

[116] Y. Zhou, B. Ni, R. Hong, M. Wang, and Q. Tian. Interaction part mining: a mid-level approach for fine-grained action recognition. In *Computer Vision and Pattern Recognition*, pages 3323–3331, 2015.

[117] Y. Zhou, B. Ni, S. Yan, P. Moulin, and Q. Tian. Pipelining localized semantic features for fine-grained action recognition. In *European Conference on Computer Vision*, pages 481–496, 2014.

[118] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *Computer Vision and Pattern Recognition*, pages 1991–1999, 2016.

[119] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *International Conference on Pattern Recognition*, pages 28–31, 2004.